Dear Editors, dear Reviewers,

Enclosed please find our manuscript entitled *Test Generation for Flow-Based Microfluidic Biochips with General Architectures*. We submit this manuscript as a regular paper for publication in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

This work builds on the method in the appended paper, which was published at the *Design, Automation and Test in Europe* Conference in 2017 as [1]. In this manuscript, we enhance the previous work as follows:

1) The model for testing Fully Programmable Valve Arrays (FPVAs) in [1] has been extended to take advantage of multiple test ports to improve test efficiency. Accordingly, the number of test patterns can be decreased by up to 50%, leading to a significant reduction of test cost. This extension is described in Section V-A and Section V-B.
2) The extended test model is capable of generating test patterns for traditional flow-based microfluidic biochips, and the number of the resulting test patterns is much smaller than that from the other previous method based on ATPG, as described in Section V-C.
3) In the extension, test of control layer leakage is covered with additional constraints and explained in detail in Section IV-C. In addition, long channels and obstacles are merged to facilitate the generation of test patterns, as described in detail in Section IV-D.
4) Furthermore, a loop relaxation technique is introduced to improve the scalability of the original model based of ILP. Constraint violations are addressed by amending the test patterns to improve the efficiency of test generation. Consequently, large designs can be dealt with using this enhanced method, as described in Section IV-A4.
5) Simulation results have been extended according to the new improvements to demonstrate their effectiveness and efficiency.

We look forward to hearing from you and thank you very much for your time and consideration.

Yours sincerely,

Chunfeng Liu, Bing Li, Bhargab B. Bhattacharya, Krishnendu Chakrabarty, Tsung-Yi Ho and Ulf Schlichtmann

Test Generation for Flow-Based Microfluidic Biochips with General Architectures

Chunfeng Liu, Bing Li, Bhargab B. Bhattacharya, Krishnendu Chakrabarty, Tsung-Yi Ho, and Ulf Schlichtmann

***Abstract*—Flow-based microfluidic biochips have become a promising platform for complex biochemical assays. As the integration of such chips is increasing, a flexible general reconfigurable platform, Fully Programmable Valve Array (FPVA), has emerged. Such a 2D array comprises regularly arranged valves using which flow-networks with different geometry, size, and connectivity can be constructed dynamically. However, test generation for such arrays becomes challenging due to the large number of potential flow-networks and transportation paths that can be configured on-chip. In this paper, we propose a strategy to generate efficient test patterns for FPVAs based on the concepts of test paths and cuts. These patterns together can cover multiple faults in both flow and control layers. We also introduce the concept of test trees and multiple cuts for a test pattern to deal with faults in FPVAs with multiple ports. Moreover, the proposed method can be applied to generate test patterns for traditional flow-based biochips with predefined architectures. Simulation results demonstrate that defects in FPVAs can be detected reliably by a limited number of test patterns generated by the proposed method. For traditional biochips with predefined architectures, these patterns also exhibit an improved test efficiency.**

I. Introduction

Microfluidic biochips have revolutionized the traditional slow and error-prone biochemical experiment flow by manipulating fluids at nanoliter level. With this miniaturization, bioassays can be scaled down and executed by moving tiny fluid samples between exact locations. Operations for manipulating fluid samples, e.g., split, move, mix, and detect, are available in these chips to implement complex bioassays. The execution of bioassays is coordinated by microcontrollers, so that a high efficiency and precision can be achieved [2]–[4]. On biochips, genomic bioassay protocols, such as nucleic-acid isolation, DNA purification, and DNA sequencing, have been demonstrated successfully. In addition, these technologies have attracted a lot of commercial attention, such as from Illumina [5] and Agilent [6].

In flow-based microfluidic biochips, microvalves are used to control the movement of fluids. The structure of a microvalve is shown in Fig. 1(a). In this structure, a flow channel is constructed on a substrate for the transportation of fluids. Above the flow channel, a control channel is constructed and connected to an air pressure. Since both channels are built from elastic materials, air pressure applied in the control channel squeezes the flow channel tightly, so that the movement of the fluid segment is blocked. If the pressure in the control channel is released, the fluid segment can resume its movement to the target device. Consequently, a valve is formed at the intersection of the two channels to control fluid movement.

Valves can also be used to build complex devices. For example, the structure of a mixer is shown in Fig. 1(b). If the three valves at the top of the mixer are actuated alternately by applying and releasing air pressure in their control channels in a given pattern similar to peristalsis, a circular flow around the device can be formed to mix different fluids. Furthermore, storage units can be constructed from normal flow channels with multiplexer-like controlling valves at each port. These units can be used to store intermediate results of operations temporarily. Fig. 1(c) shows a schematic of a mixer connected to a storage unit comprising eight cells [7].

In view of the advantages of flow-based biochips, design automation methods for them has gained much attention recently. For architectural synthesis, the method in [8] proposes a top-down flow to generate efficient biochip architectures, while the methods in [9], [10] explore the concept of distributed channel storage to improve execution efficiency. The flow channel routing problem considering obstacles is solved using an algorithm based on rectilinear Steiner minimum tree in [11]. To avoid contamination in executing operations, path searching is used in [12] to generate washing solutions for devices and channel segments. Control logic synthesis is investigated in [13]–[15] to reduce the complexity of control layer for switching valves. In addition, pressure-propagation delay in the control layer is minimized in [16] to reduce the response time of valves and synchronize their actuations. Furthermore, flow layer and control layer codesign is investigated in [17] to achieve valid routing results on both layers, and length-matching in routing control channels is investigated in [18]. Moreover, fault models of manufacturing defects and an ATPG-based test strategy for flow-based biochips are proposed in [19], [20]. Fault localization and design-for-testibility for flow-based biochips have been addressed in [21], [22].

The chip structure demonstrated in Fig. 1(c) is designed by researchers manually using preliminary tools such as AutoCAD to draw the channels at different layers. With the advances of manufacturing technologies, many thousands of valves can already be integrated into a single chip. It is thus

A preliminary version of this paper was published as [1] in the Proceedings of the Design, Automation and Test in Europe (DATE) conference, 2017. The improvements include an acceleration technique with loop relaxation, test of leakage in the control layer and test of chips with multiple pressure sensors.

Chunfeng Liu, Bing Li and Ulf Schlichtmann are with the Chair of Electronic Design Automation, Technical University of Munich (TUM), Munich 80333, Germany (e-mail: chunfeng.liu@tum.de; b.li@tum.de; ulf.schlichtmann@tum.de).

Bhargab B. Bhattacharya is with the Indian Statistical Institute, Kolkata, India (e-mail: bhargab@isical.ac.in).

Krishnendu Chakrabarty is with the Department of Electrical and Computer Engineering and the Department of Computer Science, Duke University, Durham, NC, USA (e-mail: krish@ee.duke.edu).

Tsung-Yi Ho is with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: tyho@cs.nthu.edu.tw).

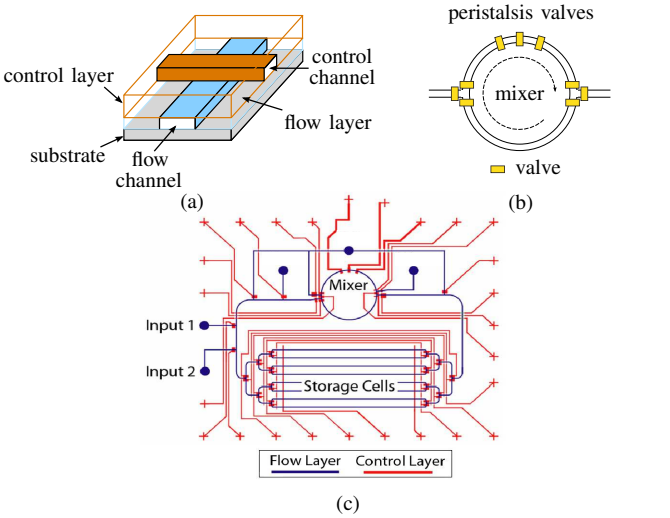


Fig. 1. Components and structure of flow-based biochips. (a) Microvalve structure. (b) Mixer. (c) Biochip with eight storage cells [7].

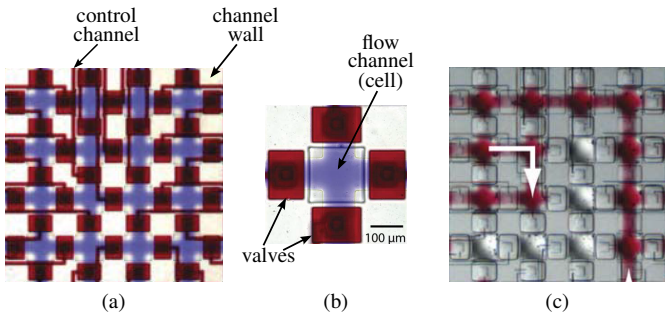


Fig. 2. Fully programmable valve array (FPVA) [23]. (a) Architecture. (b) Valve and cell. (c) Flow path construction.

challenging to design a whole biochip with the irregular structure shown in Fig. 1(c). Consequently, fully programmable valve arrays (FPVAs) have emerged with a regular structure to make the large number of valves and channels available to bioassays [2], [23]. Fig. 2(a) shows a part of the large FPVA demonstrated in [23]. A drawing of partial enlargement of four valves controlling the four directions of the fluid segment in an enclosed fluid cell is shown in Fig. 2(b). In this architecture, valves (solid blocks) are arranged regularly along horizontal and vertical flow channels (light color). These valves are controlled by air pressure through the control channels (narrow channels). By opening two valves and closing the other two, the fluid segment inside a cell can be moved to the intended direction. Consequently, flexible flow paths can be formed by opening and closing a set of valves, as shown in Fig. 2(c).

Besides flow paths, devices such as mixers can also be constructed on an FPVA, taking advantage of its flexibility and reconfigurability. For example, a 4×2 mixer and a 2×4 mixer can be constructed as shown in Fig. 3(a) and Fig. 3(b), respectively. In such a dynamic mixer, the eight valves along the enclosed channel function as peristalsis valves, which switch in a given pattern to drive the fluid segment inside the channel. Compared with the traditional mixer shown in Fig. 1(b), these dynamic mixers have different shapes and more peristalsis valves, eight in each case, to form a circular

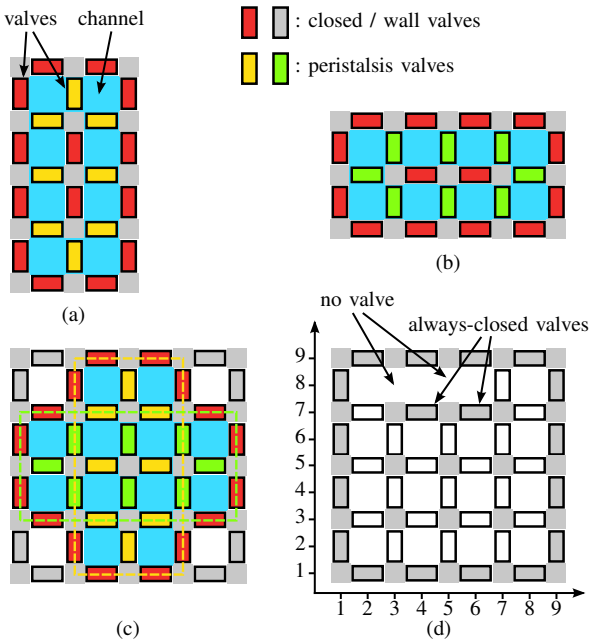


Fig. 3. FPVA with dynamic devices. (a) A 4×2 dynamic mixer. (b) A 2×4 dynamic mixer. (c) Dynamic mixers of different orientations sharing the same area. (d) FPVA with a long channel and always-closed valves (obstacles). The x-axis and y-axis show the coordinates of the valves and cells.

flow for mixing. Furthermore, the two mixers in Fig. 3(a) and 3(b) can share the same chip area as shown in Fig. 3(c) if the two mixers are not used at the same time, providing more flexibility to schedule operations on such a chip. The videos shown in [24], [25] demonstrate real cases of dynamic device mapping and fluid transportation on an FPVA.

FPVAs have a significant advantage for large-scale integration due to their regular structure. The ability of dynamic reconfigurability gives them the convenience to execute nearly any bioassays, provided specific devices such as filters and heaters are also built in the chips. To facilitate the application of FPVAs, the method in [26], [27] explores dynamic mapping of operations to reduce the maximum number of switching activities of valves, so that valve wearing can be balanced across all the valves in a chip. In [28], flow routing considering pressure-routes in establishing transportation paths on an FPVA is investigated to achieve a better assay completion time. In addition, a close-to-optimal physical design solution can also be achieved by adapting the formulation based on satisfiability in [29].

In the manufacturing process of FPVAs, defects may appear at valves and in flow and control channels. Therefore, an efficient design automation method is also required to generate test patterns for identifying chips with defects. To reduce test cost, the number of these test patterns should be as small as possible. In this paper, we address this test generation problem for FPVAs with the following contributions:

- The first systematic formulation of test strategy with test paths and cuts is proposed to enable efficient test of FPVAs after manufacturing.
- The generation of test patterns considers fault masking to cover multiple faults in the flow layer and the control

layer.
- Leakage in control channels is covered by additional test paths that are orthogonal to the test paths for stuck-at-0 faults.
- Multiple ports of FPVAs are taken advantage of to improve test efficiency.
- The proposed test framework is completely compatible with existing test framework for traditional flow-based biochips, so that no additional cost of the test platform is incurred.
- When adapted to test traditional biochips, the number of test patterns is significantly smaller than that from previously methods based on ATPG, leading to a higher test efficiency.

The rest of this paper is organized as follows. In Section II we review the state of the art of testing traditional flow-based biochips and formulate the test problem for FPVAs. In Section III we present the general test strategy for FPVAs. In Section IV, the models and their implementation for generating test paths and cuts are explained in detail. In Section V, test models are extended to cover biochips with multiple ports and to adapt the proposed method to test traditional flow-based biochips. Simulation results are shown in Section VI and conclusions are drawn in Section VII.

II. Fault Model and Problem Formulation

During manufacturing of flow-based biochips, various defects may occur. For example, the flow channel under a valve may be broken and does not allow any fluid to pass, leading to a fault equivalent to the case that the valve cannot be opened. In addition, leakage may appear between neighboring flow channels, so that fluids in them may be directed to incorrect devices or mixed unexpectedly. Furthermore, if the control channel to a valve becomes broken, air pressure may not reach the valve. Consequently, this valve cannot be closed and thus causes a constant leakage. Furthermore, a leakage may also appear between two control channels, so that the valves they drive are always opened and closed together. These cases of manufacturing defects are illustrated in Fig. 4 from [19].

According to how the defects affect the behavior of a valve or a channel, typical faults can be defined as follows:

- *Broken flow channel*: Fluid cannot pass through a channel. This is equivalent to the fault that the valve at the entrance of the channel cannot be opened.
- *Leaking flow channel*: Fluid in a channel leaks to its neighboring channel. In FPVAs, this fault is similar to the case that a valve separating two cells cannot be closed.
- *Broken control channel*: Valve cannot be closed.
- *Leaking control channel*: Two valves open or close simultaneously due to the shared air pressure in the control channels.

Since the faults that valves are stuck at the always-closed or always-open states are similar to the stuck-at-0 faults and stuck-at-1 faults in digital circuits, these faults are henceforth called *stuck-at-0 faults* and *stuck-at-1 faults* for convenience.

With these fault models, test of traditional flow-based biochips has been examined in [19]. The concept of this

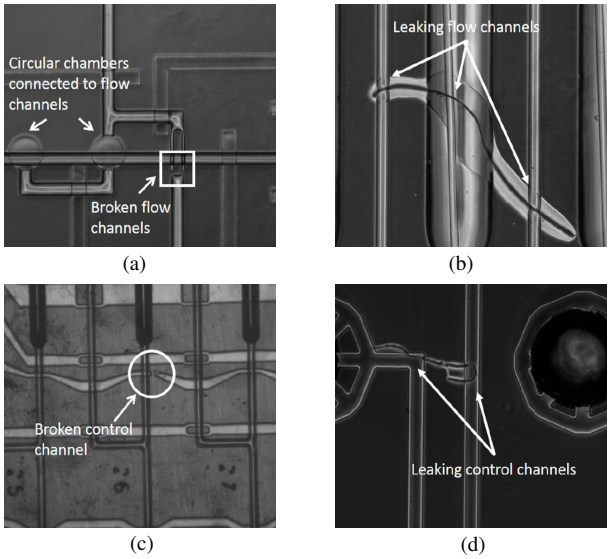


Fig. 4. Defects in flow-based biochips [19]. (a) Broken flow channel. (b) Leaking flow channel. (c) Broken control channel. (d) Leaking control channel.

method can be explained using the example illustrated in Fig. 5(a) from [19]. In this test concept, a pressure source is connected to the input port of the chip to create air pressure along the channels in the flow layer. Pressure sensors are attached to the output ports of the chip to detect air pressure. By switching the valves open or closed according to test patterns, the air pressure read by the pressure sensors at the output ports can be used to determine whether there is a fault in the chip. In this test process, an air pressure is applied to the flow channels to detect faults, so that the chip is not contaminated after test. This air pressure for the purpose of test is completely unrelated to the pressure applied in the control channels to switch valves when the chip executes bioassays.

In Fig. 5(a), an air pressure can only be detected at an output port if there is a path from the pressure source to the output port. For example, if only the valves a, g, h, i, k are open, a pressure can be detected at o_2. However, during this test if a valve on this path cannot be opened due to defects, no pressure can be detected at o_2, indicating the existence of a stuck-at-0 fault. On the other hand, if a valve on this path is also closed intentionally during the test, all paths from the source to the output ports should be blocked, so that no air pressure should be detected at any output port. If, on the contrary, the test results show that a pressure can still be observed by a sensor, a stuck-at-1 fault should exist in the chip to allow a path from the source to an output port to be formed. In this test process, the states of the valves during a pressure actuation-measurement cycle is called a *test pattern*. It is the task of test generation to generate as few test patterns as possible to detect the faults in a chip efficiently.

To generate test patterns, the method in [19] converts the biochip under test into a circuit as shown in Fig. 5(b), where the inputs of the circuit represent valves and the outputs of the circuit represent the output ports of the chip. In this circuit representation, valves along the same channel segment are inputs of AND gates, e.g., b, c, d, e, f and g, h. If two

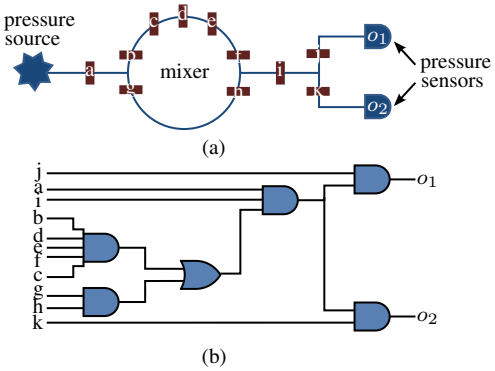


Fig. 5. Test of traditional flow-based biochips [19]. (a) Schematic of the chip under test. (b) Circuit representation of the test model for test pattern generation.

channels converge at a point, an OR gate is created in the circuit representation, since a pressure through any of these channels can reach the converging point. Consequently, the circuit represents the relation between valves and the paths from the source to the output ports in the chip. To generate test patterns for the biochip, it is equivalent to generate test patterns for the circuit representation, which can be achieved by a standard ATPG tool as shown in [19].

The ATPG-based method has the advantage that the biochip under test needs only to be converted into a circuit representation. The real test generation is performed using test generation methods for integrated circuits. However, it is challenging to apply this method directly to test FPVAs shown in Fig. 2(a). In converting a biochip into a circuit representation, the structure of the chip should be known. On an FPVA, the shapes and locations of devices and transportation channels are dynamically determined according to the operations to be executed. If the ATPG-based method is still applied, it then needs to cover a huge number of dynamic chip architectures, which is a challenging task in view of the flexibility of FPVAs.

In this paper, we propose a new test framework for detecting faults in an FPVA with only a small set of test patterns. This problem can be formulated as follows:

- Input: An FPVA architecture; the locations of long channels (no valve built, conceptually always open) and obstacles (conceptually always closed); the locations of the air pressure source and the pressure sensors.
- Output: A set of test patterns, each of which defines the open/closed states of all valves when test pressure is applied at the source and checked at the output ports by the pressure sensors.
- Objective: The number of test patterns should be as small as possible to reduce test cost; faults should be detected reliably by covering all valves.

III. Flow Paths and Cuts for FPVA Test

In the test process, the open/closed states of valves in an FPVA are set according to test patterns, and the pressure values at the output ports are measured by pressure sensors. Faults can thus be detected by comparing the readout with expected values.

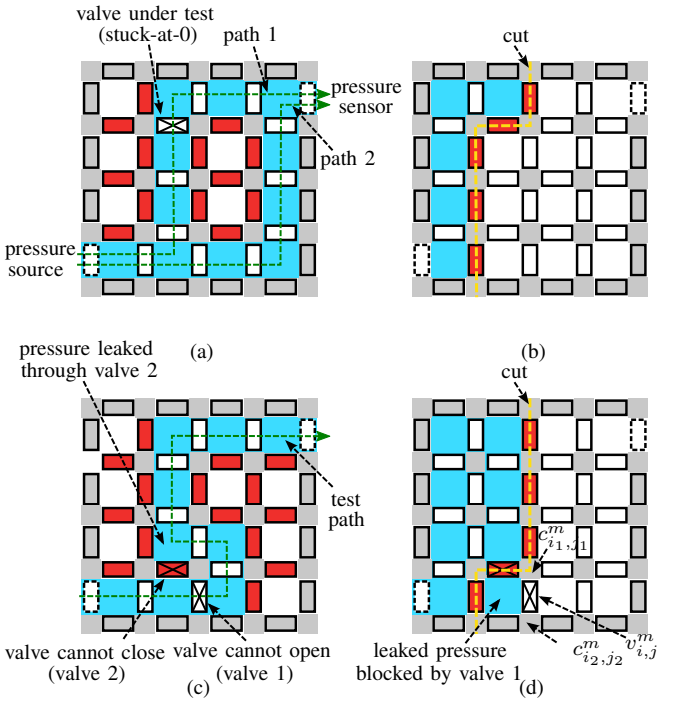


Fig. 6. Flow paths and cuts. Valves at the external boundary of the chip are always closed. (a) Test paths and stuck-at-0 fault masking. (b) Cut. (c) Test path with two-fault masking. (d) Cut with two-fault masking and variables to prevent fault masking.

To identify whether faults exist in a chip, a test pattern should ensure that faults are observable by the pressure sensors at the output ports. For example, faults caused by valves that cannot be opened, stuck-at-0 faults, can be detected by creating paths from the pressure source to the pressure sensors. If, however, no pressure sensors can read a valid pressure value at the outputs, some valves on the path should have defects, and cannot be opened by the test pattern to allow the test pressure to pass. Similarly, if a test pattern cuts off all the paths from the pressure source to the pressure sensors but the sensors still detect pressure valves, stuck-at-1 faults allowing pressure leakage at valves must exist in the chip. Fig. 6(a) and 6(b) illustrate the concepts of test patterns to test stuck-at-0 and stuck-at-1 faults.

When test patterns are applied, some faults might mask each other. For example, two test paths are created in the FPVA illustrated in Fig. 6(a) simultaneously. If a valve on one of the paths is defected, the air pressure can still reach the pressure sensor through the other path. Therefore, this stuck-at-0 fault cannot be observed at the output port. To avoid this path interference problem, only simple paths without loops should be constructed. These paths are called *test paths* henceforth. Similar to test paths, test cuts can be constructed to detect stuck-at-1 faults. A *test cut* is formed by a set of valves that separate the pressure source and the pressure sensors completely when they are closed, so that no test pressure can reach the sensors, as illustrated in Fig. 6(b).

Based on the analysis above, the test patterns that need to be constructed for testing an FPVA can be defined as follows:

- *Test-path patterns*: A set of paths between the pressure source and the pressure sensors. These paths should cover

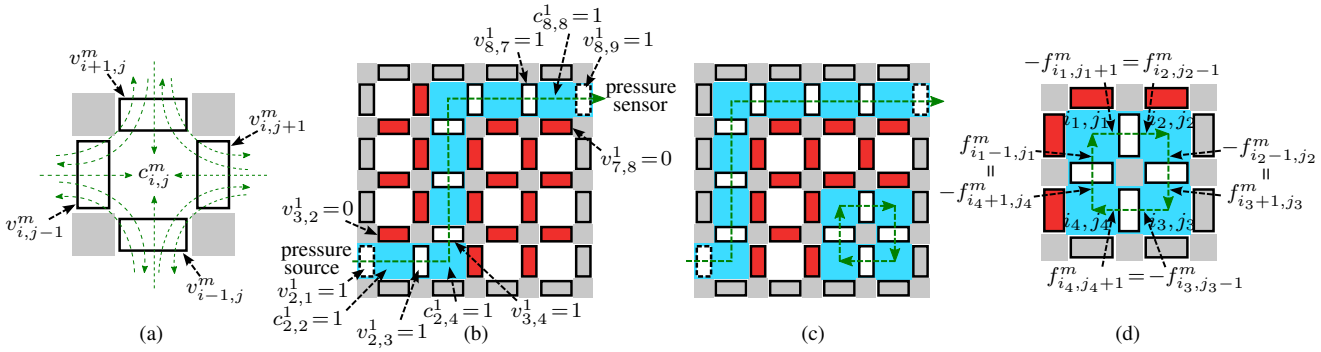


Fig. 7. Flow path model. (a) Constraint variables for valves and cells. (b) Path construction using constraints. (c) Disjoint loop. (d) Flow constraints along a disjoint loop.

each valve at least once. A test path should be a simple path without a loop, so that no parallel path segment that can mask stuck-at-0 faults is present. A test path is thus similar to a "self-avoiding walk" on a lattice, which is well studied elsewhere in the context of grid graphs [30]. In the test process, test paths are applied one after another, in each of which only the valves on the path are opened and all the other valves are closed. If no test pressure is detected, stuck-at-0 fault should exist.

- *Test-cut patterns*: A set of cuts, each of which is formed from a series of valves separating the pressure source and the pressure sensors completely when they are closed. The generated cuts should cover each valve at least once to test whether it can be closed. When a cut is applied but air pressure is still detected by a sensor, at least a valve in the cut should have stuck-at-1 fault.

The path and cut test patterns can detect any single fault in the chip. If multiple faults exist, fault masking might still happen. Assume that there are two faulty valves, one of which cannot be opened (valve 1, stuck-at-0) and the other cannot be closed (valve 2, stuck-at-1). Also assume that the test path used to test valve 1 is constructed as shown in Fig. 6(c), and the cut used to test valve 2 is constructed as shown in Fig. 6(d). In Fig. 6(c), the test pressure can still bypass the broken valve 1 and reach the pressure sensor though valve 2. In other words, the pressure leakage through valve 2 masks the stuck-at-0 fault at valve 1. In Fig. 6(d), the pressure leakage through valve 2 is blocked by valve 1, so that no test pressure can reach the sensor, leading to mutual masking of faulty valves. Consequently, none of these faulty valves can be detected by the test patterns. To solve this problem, such mutual masking cases should be excluded from the generated test patterns.

IV. Generating Test-Path and Test-Cut Patterns

The objective of generating path and cut test patterns is to minimize the number of test patterns to reduce test cost. The path patterns together should cover all valves to detect stuck-at-0 faults, and so are the cut patterns to detect stuck-at-1 faults. Furthermore, leakage in control layers causes two valves to switch simultaneously, so that such faults should also be covered by the test patterns. The test patterns described in this section assume that there is a single pressure source and a single pressure sensor for fault test. Test patterns with multiple sensors will be discussed in Section V.

A. Constructing path test patterns

In an FPVA, a test path can pass through a valve from any of the two directions. The test paths together should cover each valve in the chip at least once. In the proposed method, we describe this path generation using an Integer Linear Programming (ILP) model. The scalability of this model is further improved using a heuristic loop removal technique.

1) Path pattern formulation: In an FPVA, a fluid cell is defined as the channel area surrounded by four valves. A test path can enter such a cell and leave it from any of the four valves surrounding the cell. Consequently, there are 12 possible directions for a path passing through a cell, as illustrated by the dashed lines in Fig. 7(a).

In describing the path model, a valve and a fluid cell in an FPVA are denoted as $\mathtt{V}_i$, and $\mathtt{C}_{i,j}$, respectively, where (i,j) is the coordinate as shown in Fig. 3(d). Assume all valves can be covered by no more than n_p test paths, where n_p is a given constant. For the cell at the location (i,j), we assign a 0-1 variable $c^m_{i,j}$ to represent whether the mth path travels through the cell. If the mth path travels through the cell, $c^m_{i,j}=1$; otherwise $c^m_{i,j}=0$. For the valves at the left, right, upper and lower sides of the cell, we assign 0-1 variables $v^m_{i,j-1}$, $v^m_{i,j+1}$, $v^m_{i+1,j}$ and, $v^m_{i-1,j}$, respectively. If the mth path travels through a valve, the corresponding variable is set to 1; otherwise, it is set to 0.

If the mth path travels through the cell $\mathtt{C}_{i,j}$ at the location (i,j), this path should travel through exactly two valves that surround the cell. Consequently, the relation between the cell and the valves surrounding it can be established as

$$v^m_{i,j-1}+v^m_{i,j+1}+v^m_{i+1,j}+v^m_{i-1,j}=2c^m_{i,j}, \quad \forall\ \mathtt{C}_{i,j}\in\mathbf{C},\ m=1,2,\ldots,n_p \tag{1}$$

where $\mathbf{C}$ is the set of all the cells in the FPVA. (i,j) is the coordinate of the cell $\mathtt{C}_{i,j}$. n_p is the maximum number of the test paths.

To initiate a test path, we set the variable $v^m_{i,j}$ of the valve and the variable $c^m_{i,j}$ of the cell that are connected to the pressure source always to 1. Similarly, the variables for the valve and the cell connected to the pressure sensor are initialized. The constraint (1) forces two valves neighboring

a cell to appear on a test path, and if a valve appears on a path, two cells neighboring it must also appear on the path. This chaining effect of the variables propagates further until the pressure sensor is reached, thus defining a whole test path from the pressure source to the sensor. Fig. 7(b) shows a partial example of this chaining propagation defined by (1).

To guarantee that a valve is covered at least once by the test paths, one of the constraint variables $v_{i,j}^m$ for the valve $\mathtt{V}_{i,j}$ out of the n_p paths must be 1, leading to

$$\sum_{m=1}^{n_p} v_{i,j}^m \geq 1, \ \forall \ \mathtt{V}_{i,j} \in \mathbf{V} \tag{2}$$

where $\mathbf{V}$ is the set of all the valves in the FPVA and (i,j) is the coordinate of the valve $\mathtt{V}_{i,j}$.

2) Excluding disjoint loops from test paths: With the constraints (1) and (2), disjoint loops may appear on a test path. For example, these constraints do not prevent the disjoint loop at the lower right side of the FPVA in Fig. 7(c) from happening. All the valves and cells on this loop meet the constraints (1) and (2), but this loop leads to a false valve coverage in test, because test pressure from the source cannot reach any valve on this loop to verify whether it can be opened.

The disjoint loop can be removed by forcing a flow from the pressure source to any segment of the path. Assume that the pressure source needs to provide one unit of pressure volume to fill a fluid cell, the total pressure volume stored on a path should be equal to the number of fluid cells on the path. If the path is not a single path but contains a disjoint loop, the number of fluid cells on the path should be larger than the pressure volumes from the source, since no pressure volume can reach the cells on the loop.

When applying a test path onto an FPVA, the pressure source provides pressure volumes that flow through the path. To represent the pressure volume passing through a valve at the location (i,j), we define an integer variable $f_{i,j}^m$. This variable is positive when viewed from a cell which the pressure flow enters; it is negative when viewed from a cell which the pressure flow leaves. In addition, the pressure propagation can pass through a valve only if the valve is on the test path, under the condition $v_{i,j}^m = 1$. Otherwise, $f_{i,j}^m$ must be set to 0. This condition constrains $f_{i,j}^m$ as

$$f_{i,j}^m \leq v_{i,j}^m \cdot \mathcal{M} \quad \text{and} \quad f_{i,j}^m \geq -v_{i,j}^m \cdot \mathcal{M} \tag{3}$$

where $\mathcal{M}$ is a large positive constant [31].

Since a fluid cell is surrounded by four valves in general, the pressure volume stored in a cell at the location (i,j) when the mth test path is applied is equal to the sum of the volumes flowing through the four surrounding valves. This relation can be written as

$$f_{i,j-1}^m + f_{i,j+1}^m + f_{i+1,j}^m + f_{i-1,j}^m = c_{i,j}^m \tag{4}$$

where the valves on the left of, on the right of, above and below the cell $\mathtt{C}_{i,j}$ at the location (i,j) are indexed by $(i,j-1)$, $(i,j+1)$, $(i+1,j)$ and $(i-1,j)$, respectively.

Constraint (4) is capable of preventing disjoint loops from appearing effectively. Assume there is a disjoint loop on the mth path and the cells on the disjoint loop are $\mathtt{V}_{i_1,j_1}, \mathtt{V}_{i_2,j_2}, \ldots \mathtt{V}_{i_l,j_l}$, where the valves $\mathtt{V}_{i_1,j_1}$ and $\mathtt{V}_{i_l,j_l}$ are neighbors to form a loop. For each cell on the loop, a constraint in the form of (4) is created. Adding the left and right sides of these constraints together, we have

$$\sum_{(i,j)\in I_l} (f_{i,j-1}^m + f_{i,j+1}^m + f_{i+1,j}^m + f_{i-1,j}^m) = \sum_{(i,j)\in I_l} c_{i,j}^m \tag{5}$$

where I_l is the index set $\{(i_1,j_1),(i_2,j_2),\ldots(i_l,j_l)\}$ for the cells on the loop. On a disjoint loop, the sum on the left side of (5) is always equal to 0, because no pressure flow enters the loop. This contradicts the fact that $\sum_{(i,j)\in I_l} c_{i,j}^m$ is always larger than 0 if the cells are on the test path. Therefore, constraint (4) for each cell guarantees that all test paths are simple paths without disjoint loops to avoid false coverage during test. The concept of this model is illustrated in Fig. 7(d).

3) Finding the minimum set of test paths: The path constraints defined above assume that the number n_p of test paths to cover all valves are known. In our formulation, we first set n_p to a constant and then find a set of paths whose number is no larger than n_p to cover all valves. For each path, we assign a 0-1 variable p_m, $1 \leq p_m \leq n_p$ to indicate whether this path is used. Because any valve on the p_mth path marks the path to be used, p_m can be constrained as

$$p_m \cdot \mathcal{M} \geq \sum_{(i,j)\in I} v_{i,j}^m \tag{6}$$

where I is the index set of all valves. $\mathcal{M}$ is a positive constant larger than the number of valves in the array. If a valve is on the mth path, the right side of (6) is larger than 0, so that p_m must be 1 to meet the constraint.

With the constraints above, the ILP problem to find a minimum set of test paths that cover all valves can be formulated as follows

$$\text{minimize} \quad \sum_{m=1}^{n_p} p_m \tag{7}$$

$$\text{subject to} \quad (1)-(4) \text{ and } (6). \tag{8}$$

Since the number of paths n_p is specified as a constant in the formulation, it is possible that the ILP problem above has no solution, meaning that not all the valves can be covered by n_p test paths. If this happens, we increase n_p and solve the optimization problem again.

4) Relaxing constraints for excluding disjoint loops and heuristic path reconstruction: To exclude disjoint loops from test paths, constraint (4) is created for each cell and included in the optimization problem (7)–(8). These strict constraints, however, lead to huge computation overhead. To address this issue, we introduce a technique to relax these strict constraints to accelerate the computing process.

The basic idea of this technique is similar to Lagrangian relaxation, in which a part of the constraints are moved into the objective and violations of these constraints are punished by weights. The solution of a relaxed problem approximates the solution of the original problem, though violations of the original constraints may not be avoided completely. In the proposed framework, we deal with these constraint violations by amending the solution heuristically.

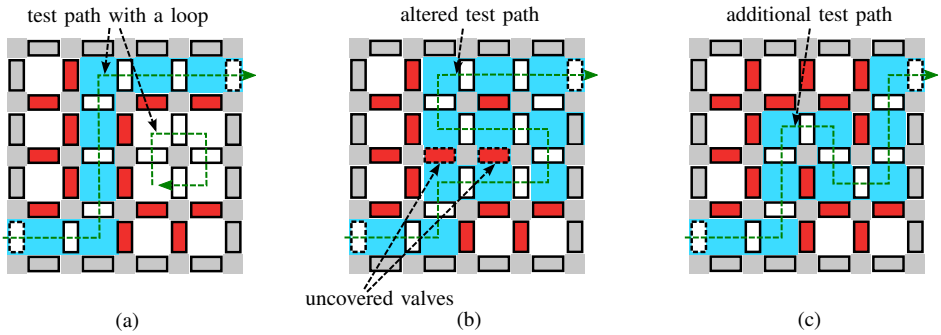


Fig. 8. Eliminating a disjoint loop. (a) A test path containing a disjoint loop. (b) Altered test path partially covering valves on the loop. (d) An additional test path created to cover the rest valves on the loop.

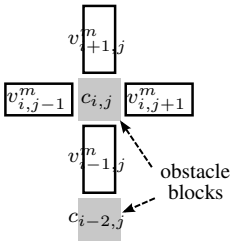


Fig. 9. Constraint variables for cut-set modeling.

In the disjoint loop constraint (4), the sum of the pressure volume going through the loop must be equal to the number of the cells. To relax this constraint, we allow this sum to be smaller than the number of cells, as

$$f^m_{i,j-1}+f^m_{i,j+1}+f^m_{i+1,j}+f^m_{i-1,j}\leq c^m_{i,j}. \tag{9}$$

To prevent disjoint loops from happening, we minimize the difference between the left side and the right side of (9). This difference is written as

$$c_{i,j}-(f^m_{i,j-1}+f^m_{i,j+1}+f^m_{i+1,j}+f^m_{i-1,j})=l^m_{i,j}. \tag{10}$$

Accordingly, the path generation problem (7)–(8) can be relaxed as

$$\text{minimize} \quad \alpha\cdot\sum_{m=1}^{n_p} p_m+\beta\cdot\sum_{m=1}^{n_p}\sum_{\mathtt{c}_{i,j}\in\mathbf{C}} l^m_{i,j} \tag{11}$$

$$\text{subject to} \quad (1)-(3),(6),(9)-(10) \tag{12}$$

where α is set to 10 times of β to give priority to the lower number of test patterns.

Since the ILP model (11)–(12) only minimizes the occurrence of disjoint loops, such loops may still appear in the result returned by the solver. An example of this case is shown in Fig. 8(a). To cover the cells on the disjoint loop, we split these cells into two parts. The first part is merged with the original simple path to construct a path from the pressure source to the pressure sensor, as shown in Fig. 8(b), where two valves are left uncovered. These valves are merged into a new path modified from the original simple path to increase fault coverage, as shown in Fig. 8(c). In implementing this post-processing step, we break all the disjointed loops resulted by solving (11)–(12) and merge them into existing simple paths together to reduce the number of newly created test paths.

B. Constructing cut test patterns

After generating the test paths, we also need to create cuts to test whether all valves can be closed. In this test, if all valves in a cut are closed correctly, there should be no test pressure going from the pressure source to the sensor. Because a cut separates the pressure source and the pressure sensor, the two ends of a cut must touch the boundary of the chip, as shown in Fig. 6(b). Observing this phenomenon, we search the valves along the boundary of the chip in two directions starting from the pressure source until the pressure sensor is reached from both directions. The resulting valves are added into two sets, and a cut must include a valve from each of these sets. Therefore, the problem to find cuts can be reformulated as follows:

Cut generation: Find a minimum number of simple paths without loops to form cuts that cover each valve in the chip at least once. Each cut must have a valve from each of the two valve sets identified by searching along the external boundary of the chip from the pressure source to the pressure sensor.

The problem above is a complementary problem of finding a set of test paths covering all valves described in Section IV-A. To solve this problem, we can still use the formulation (11)–(12) together with the constraint that a cut must contain a valve in each of the two sets found by searching along the chip boundary. In this complementary formulation, constraint variables are not assigned to cells but to the obstacle blocks between valves to form cuts. This variable assignment is illustrated in Fig. 9.

As discussed in Section III, we must prevent the scenario shown in Fig. 6(c) and (d) from happening. Otherwise, two faulty valves mask each other and thus cannot be detected. This scenario can be described as that a new cut can be formed by only one new valve with other valves from an existing cut. Assume there is a valve $\mathtt{v}_{i,j}$ and the two obstacle blocks at the two ends of the valve are indexed by (i_1,j_1) and (i_2,j_2). To prevent fault masking as illustrated in Fig. 6(c) from being formed, we add an additional constraint to the optimization problem as

$$c^m_{i_1,j_1}+c^m_{i_2,j_2}-1\leq v^m_{i,j}. \tag{13}$$

This constraint informs the solver that if the two ends of a valve are in a cut, this valve must be included in the cut. Consequently, it is not possible anymore that another single valve forms a new cut with some valves in the current cut, so that the single-fault masking problem can be avoided.

C. Testing control layer leakage

If there is a leakage from a control channel to another control channel, an air pressure in this control channel to close a valve also closes the other valve simultaneously. Such faults usually happen at adjacent valves during manufacturing [19].

In the test paths described in Section IV-A, control layer leakage has actually been covered in part. For example, on

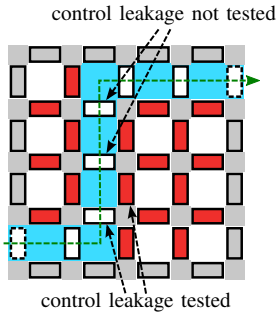


Fig. 10. Partial control leakage test with paths.

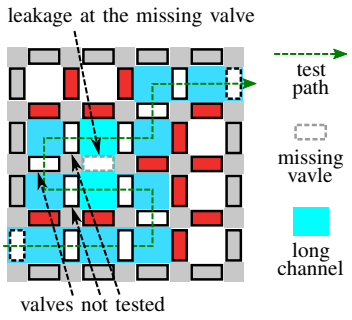


Fig. 11. Test path through a long channel twice. The missing valve causes bypassing of valves on the path.

the test path in Fig. 10, the valves that form the boundary of the path are closed while the valves along the path are opened. If the control pressure to a closed valve at the boundary of the path leaks into the control channel of an opened valve on the test path, the test path is then blocked. Consequently, no test pressure can be detected by the sensor, indicating a fault in the chip.

Besides the valve pairs above, there are further valve pairs whose control channel leakage should be tested. For example, the valves along the path in Fig. 10 need to be checked, because these valves are opened together during the test, so that the case a valve is closed and another is opened is not covered. To deal with these untested valve pairs, we generate further paths going through only one of the valves in an untested valve pair along a path, while the other valve is kept closed. In other words, only one of the valves in such a pair appears on a new test path.

To generate new test paths to test control channel leakage, we first scan existing flow paths generated as described in Section IV-A to identify untested valve pairs. Assume that the valves at (i_1, j_1) and (i_2, j_2) need to be tested for control channel leakage. We force one of them to stay open and the other to be closed when new test paths are generated. As defined in Section IV-A, the 0-1 variable $v_{i,j}^m$ represents whether a valve is on the mth path. Therefore, the additional constraint can be written as

$$v_{i_1,j_1}^m + v_{i_2,j_2}^m = 1 \tag{14}$$

which can be included in the optimization problem (11)–(12) to cover control leakage.

D. Dealing with long channels and obstacles

An FPVA may contain long channels and obstacles to enable certain functionalities such as allowing special devices to be built into the chip. In long channels, some valves are missing. When constructing a test path, all the valves on the path are opened and the other valves are closed to form the boundary of the path. However, missing valves and obstacles may cause special problems in the test. For example, the flow path in Fig. 11 runs through a long channel twice. Since the missing valve in the long channel cannot be closed, the short path through the missing valve leaks the test pressure. Accordingly, the sensor can still detect a test pressure, even if some valves

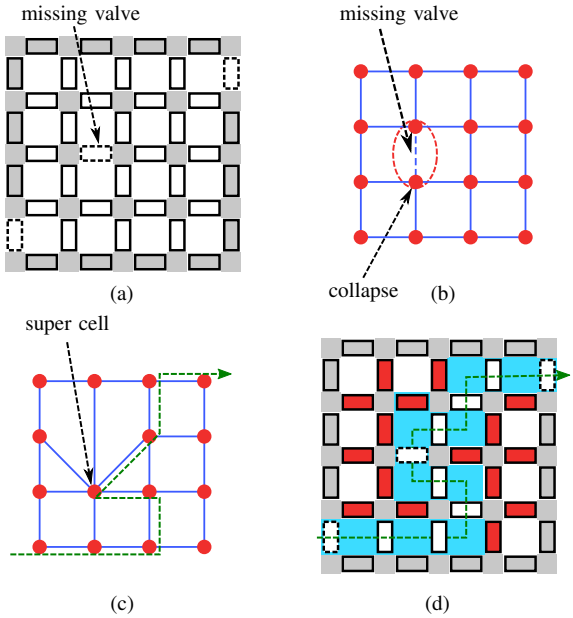


Fig. 12. Generating test paths on an FPVA with a long channel. (a) The original FPVA. (b) Converting the FPVA into a connection graph. (c) Cells in a long channel are collapsed into a super cell. Flow paths are constructed on the new connection graph. (d) A corresponding flow path on the original FPVA.

on the left side of the long channel cannot be opened due to manufacturing defects. Consequently, the test path become invalid, although if meets the constraints described previously.

To address the issues above, we introduce the concept *super cell*. Fig. 12(a) is the FPVA with a long channel. In Fig. 12(b), the FPVA is converted to a connection graph to show the relationship between cells and valves, where nodes represent the cells and edges represent the valves. Because the cells in the long channel are always connected together, we can collapse them into one super cell. All the valves surrounding the long channel then surround this new super cell. The new connection graph of is shown in Fig. 12(c). We can then apply the ILP model (11)–(12) on the new connection graph to construct test paths. Because all the cells inside a long channel are treated as one cell, the corresponding paths do not run through the long channel more than once. Fig. 12(d) shows one of the actual test paths on the original FPVA.

The concept of collapsing nodes in the connection graph is also applied to generate cuts while considering obstacles efficiently. Instead of representing cells, the nodes in the connection graph then represent the small obstacle blocks between valves as shown in Fig. 9 to cut off all paths from the pressure source to the sensors.

V. Test of FPVAs with Multiple Sensors

In the previous sections, we have discussed the strategy to test FPVAs using a single pressure source and a single pressure sensor. In practice, an FPVA may have more than 2 ports which can be connected to pressure sources or pressure sensors. Taking advantage of these testing ports can improve the testing efficiency potentially. In the formulation discussed in the following, we assume that only one pressure source and multiple pressure sensors are used. If more than one

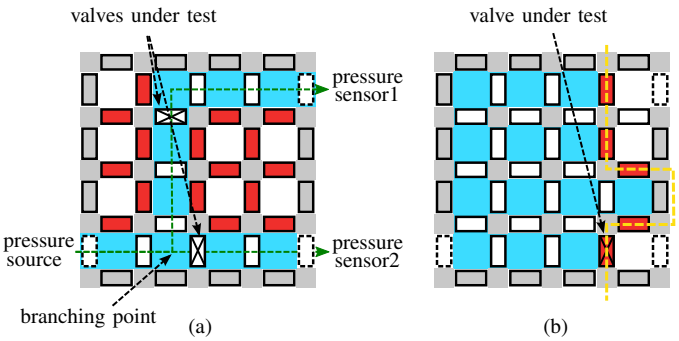


Fig. 13. Multiple-port test. (a) Test pattern for stuck-at-0 faults. (b) Test pattern for stuck-at-1 faults.

pressure source exists, either the air pressure generated by these sources would affect each other, or the test problem is split into separate blocks, in each of which only one pressure source appears.

The scenario of testing an FPVA with one pressure source and two pressure sensors is shown in Fig. 13. In Fig. 13(a) a test pattern for testing stuck-at-0 faults is shown. Due to the available multiple pressure sensors, a branching point appears on this test pattern, leading to a *test tree* to test stuck-at-0 faults. During the test, all the valves on this test tree are opened and the valves on the boundary of this tree are closed. If no pressure is detected at one of the pressure sensors, a stuck-at-0 fault exists at a valve on this tree. Similarly, the test pattern used to detect stuck-at-1 faults is shown in Fig. 13(b), where all the valves on the walls are closed. If one of the sensors detects a pressure, at least a valve on the wall is broken, exhibiting a stuck-at-1 fault.

A. Multiple-port test of stuck-at-0 faults

Similar to building test paths for testing stuck-at-0 faults in an FPVA with a single pressure source and a single pressure sensor, test trees can be built to test stuck-at-0 faults in an FPVA with multiple ports to improve test efficiency. The root of a test tree is at the pressure source and the leaf nodes are at the pressure sensors. On such a test tree, if all the valves work properly, the opened valves allow the test pressure to be conducted from the pressure source to all the pressure sensors. If a valve cannot be opened, it then blocks the air pressure to at least one of the pressure sensors. To test all the valves in an FPVA for stuck-at-0 faults, multiple trees are needed to cover each valve in the chip at least once. Since the tree structure is more complex than the simple path in the test of a single-source single-sensor FPVA, the constraints described in Section IV-A need to be extended accordingly, described as follows.

In the formulation for constructing simple test paths in Section IV-A, the 0-1 variable $c_{i,j}^m$ represents whether the mth path travels through the cell at the location (i,j). The 0-1 variables $v_{i,j-1}^m$, $v_{i,j+1}^m$, $v_{i+1,j}^m$ and, $v_{i-1,j}^m$ represent whether the mth path travels through the valves at the left, right, upper and lower sides of the cell at the location (i,j), as illustrated in Fig. 7(a). Constraint (1), however, only allows a path to go through a cell without branching, since only two valves

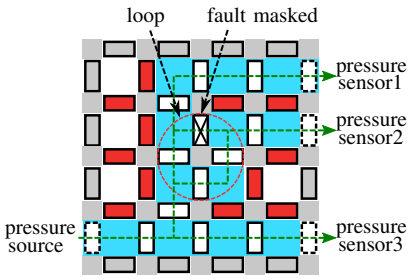


Fig. 14. Loop on test tree. A defected valve on the loop cannot be detected.

surrounding a cell can be covered under this constraint. At a branching point of a test tree, the number of these valves can actually exceed two, as shown in Fig. 13(a). Accordingly, constraint (1) should be modified as

$$v_{i,j-1}^m + v_{i,j+1}^m + v_{i+1,j}^m + v_{i-1,j}^m \geq 2 - (1 - c_{i,j}^m) \cdot \mathcal{M} \quad (15)$$

$$v_{i,j-1}^m + v_{i,j+1}^m + v_{i+1,j}^m + v_{i-1,j}^m \leq c_{i,j}^m \cdot \mathcal{M} \quad (16)$$

$$\forall\ \mathtt{c}_{i,j} \in \mathbf{C},\ m = 1, 2, \ldots, n_p$$

where $\mathbf{C}$ is the set of all the cells in the FPVA. (i,j) is the coordinate of the cell $\mathtt{c}_{i,j}$. $\mathcal{M}$ is a large positive constant. (15) corresponds to the case that the cell at the location (i,j) is covered by the mth test tree and (16) the case this cell is not covered.

Similar to (1), the constraints (15)–(16) may cause disjoint loops to appear. To exclude these loops, the constraints described in Section IV-A2 should still be included for the test scenario with multiple pressure sensors. In addition, the constraints (15)–(16) allow more than 2 valves surrounding a cell to be covered by a test tree. This very relaxed formulation may cause further loops between the branches of the trees, as shown in Fig. 14. On this loop, a defected valve cannot be detected since the test pressure can always be propagated through the other part of the loop.

To exclude loops described above from a test tree, the relation between the numbers of the valves and the cells on the flow tree can be specified. Since the cells can be viewed as nodes in a graph and the valves the edges connecting them, the number of valves on a test tree is always one smaller than the number of cells. Therefore, we specify the following constraint to exclude the loops incurred by (15)–(16), as

$$\sum_{\mathtt{c}_{i,j} \in \mathbf{C}} c_{i,j}^m - \sum_{\mathtt{v}_{i,j} \in \mathbf{V}} v_{i,j}^m = 1, \quad m = 1, 2, \ldots, n_p \quad (17)$$

where the sum operations result in the numbers of valves and cells on the mth test tree, respectively.

After extending the constraints for test trees, the coverage of all valves and the minimization of the number of test trees can be achieved by adapting (11)–(12).

B. Multiple-port test of stuck-at-1 faults

In Section IV-B, we have described a method to generate cuts in an FPVA to test whether all valves can be closed properly. A cut separates the pressure source and the pressure sensor so that no test pressure should be detected by the sensor. Otherwise, a stuck-at-1 fault must exist on a valve in the cut.

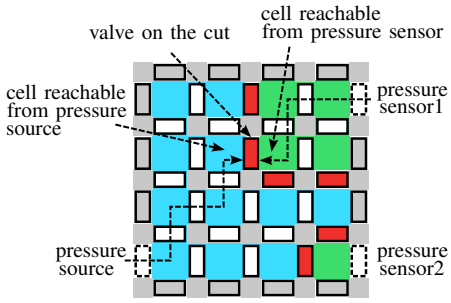


Fig. 15. Test stuck-at-1 faults with cut and multiple pressure sensors.

When multiple pressure sensors are used, the method should be extended accordingly, though the basic test concept remains unchanged.

The scenario of testing stuck-at-1 faults with multiple sensors is illustrated in Fig. 15, where the cut is composed of multiple segments to separate the pressure source from all the pressure sensors. When any valve on the cut has a stuck-at-1 fault so that it cannot be closed, the leaked air pressure must be able to reach one of the pressure sensors to guarantee a fault detection. This is equivalent to the formulation that a valve on the cut should be reachable from the pressure source and from one the pressure senors backwards simultaneously.

In an FPVA, a valve $\mathtt{V}_{i,j}$ at the location (i,j) is adjacent to two cells. According to the orientation of the valve, the adjacent cells can be denoted as $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ or $\mathtt{C}_{i-1,j}$ and $\mathtt{C}_{i+1,j}$, respectively. For simplicity, we only use $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ to describe the reachability conditions to test valves on the cut. During test, if $\mathtt{V}_{i,j}$ is on a cut, at least a path from the pressure source should reach one and only one of the cells $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ to guarantee the test pressure can reach one side of $\mathtt{V}_{i,j}$. To guarantee a leaked pressure to be detected, there should be at least a path from one of the pressure sensors to reach the other cell neighboring $\mathtt{V}_{i,j}$.

Assume that the 0-1 variables $s_{i,j-1}$ and $s_{i,j+1}$ represent whether there are paths from the pressure source to the cells $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$, respectively, and that the 0-1 variables $t_{i,j-1}$ and $t_{i,j+1}$ represent whether there are paths from the pressure sensors to the cell $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$, respectively. The constraints that $\mathtt{V}_{i,j}$ is on a cut can be expressed as

$$s_{i,j-1}+t_{i,j-1}-(1-v_{i,j})\cdot\mathcal{M}\leq 1 \quad (18)$$
$$s_{i,j-1}+t_{i,j-1}+(1-v_{i,j})\cdot\mathcal{M}\geq 1 \quad (19)$$
$$s_{i,j+1}+t_{i,j+1}-(1-v_{i,j})\cdot\mathcal{M}\leq 1 \quad (20)$$
$$s_{i,j+1}+t_{i,j+1}+(1-v_{i,j})\cdot\mathcal{M}\geq 1 \quad (21)$$
$$s_{i,j-1}+s_{i,j+1}-(1-v_{i,j})\cdot\mathcal{M}\leq 1 \quad (22)$$
$$t_{i,j-1}+t_{i,j+1}-(1-v_{i,j})\cdot\mathcal{M}\leq 1 \quad (23)$$

where $v_{i,j}$ represents whether the valve $\mathtt{V}_{i,j}$ is on the cut. The large constant $\mathcal{M}$ ensures that these constraints are only valid when $v^m_{i,j}=1$.

If $v^m_{i,j}=1$, only one of the variables from $s_{i,j-1}$ and $t_{i,j-1}$ or from $s_{i,j+1}$ and $t_{i,j+1}$ can be one, so that the cells $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ can only be reached from either the pressure source or the pressure sensors, but not both. The constraints (22) and (23) exclude the case that the valve $\mathtt{V}_{i,j}$ is completely located on a side of a cut.

The constraints above guarantee that all the valves meeting these constraints together form a cut separating the pressure source and the pressure sensors completely. If this would not be the case, air pressure can thus travel from the pressure source to the pressure sensors freely, so that it can reach the adjacent cells $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ simultaneously. Consequently, the left side of the equations (18) and (20) would be 2, thus contradicting the constraints.

To specify the reachability of the cells $\mathtt{C}_{i,j-1}$ and $\mathtt{C}_{i,j+1}$ from the pressure source or the pressure sensors, the concept of pressure flow described in Section IV-A2 can be applied. In the general case, a variable $f^s_{i,j}$ is used to denote the pressure flow originating from the pressure source and running through a valve or a cell at the location (i,j). If the total flow running through all the valves surrounding a cell $\mathtt{C}_{i,j}$ is larger than 1, this cell is reachable from the pressure source, formulated as follows,

$$f^s_{i,j-1}+f^s_{i,j+1}+f^s_{i+1,j}+f^s_{i-1,j}\geq s_{i,j},\quad \forall \mathtt{C}_{i,j}\in\mathbf{C} \quad (24)$$

where $\mathbf{C}$ is the set of all the cells in the FPVA.

Additionally, a cell $\mathtt{C}_{i,j}$ has to be reachable if one of its neighboring cells is reachable and they are not separated by a valve on the cut, formulated as follows,

$$s_{i,j-2}-v_{i,j-1}\cdot\mathcal{M}\leq s_{i,j} \quad (25)$$
$$s_{i,j+2}-v_{i,j+1}\cdot\mathcal{M}\leq s_{i,j} \quad (26)$$
$$s_{i-2,j}-v_{i-1,j}\cdot\mathcal{M}\leq s_{i,j} \quad (27)$$
$$s_{i+2,j}-v_{i+1,j}\cdot\mathcal{M}\leq s_{i,j}. \quad (28)$$

Furthermore, to ensure no pressure is allowed to pass through a valve $\mathtt{V}_{i,j}$ on a cut, we need to specify that $f^s_{i,j}=0$ if the valve $\mathtt{V}_{i,j}$ is on the cut, as

$$f^s_{i,j}-(1-v_{i,j})\cdot\mathcal{M}\leq 0 \quad (29)$$
$$f^s_{i,j}+(1-v_{i,j})\cdot\mathcal{M}\geq 0. \quad (30)$$

The constraints (24)–(30) describe the case for pressure propagation from the pressure source. Similar constraints should be added to describe the condition of pressure propagation from the pressure sensors. These constraints expand the formulation in Section IV-B to form cuts composed of multiple segments for stuck-at-1 fault test.

C. Applying multiple-port test onto traditional biochips

The work [19] proposes a method for testing traditional continuous-flow microfluidic biochips. This method converts a biochip architecture into a digital circuit representation. Afterwards, an ATPG tool is adopted to generate test patterns. To apply this method onto FPVAs is, however, very challenging, because of the complexity of converting the dynamic architectures of FPVAs into a circuit representation. On the contrary, our method based on test paths/trees and cuts solves this problem by exploring the relation between valves and cells in the chip directly, so that it can deal with any chip architectures efficiently. Accordingly, this method can also be

TABLE I
RESULTS OF TEST PATTERN GENERATION WITH SINGLE PRESSURE SOURCE AND SINGLE PRESSURE SENSOR

FPVA		Direct Approach							Loop Acceleration						
Dimension	n_v	n_p^d	$t_p^d(s)$	n_c^d	$t_c^d(s)$	n_l^d	$t_l^d(s)$	N^d	n_p^a	$t_p^a(s)$	n_c^a	$t_c^a(s)$	n_l^a	$t_l^a(s)$	N^a
5 ×5	40	2	0.02	8	0.67	3	0.01	13	2	0.02	8	0.16	3	0.02	13
10×10	380	2	0.72	18	46	3	0.04	23	2	2.61	18	12	3	0.03	23
15×15	420	2	48	28	984	4	0.12	34	2	78	28	40	3	73	33
20×20	760	2	758	*	*	*	*	*	2	363	38	146	3	163	43
25×25	1200	*	*	*	*	*	*	*	3	1200	48	472	3	966	54
30×30	1740	*	*	*	*	*	*	*	3	1201	58	1648	4	1243	65

* – No valid result

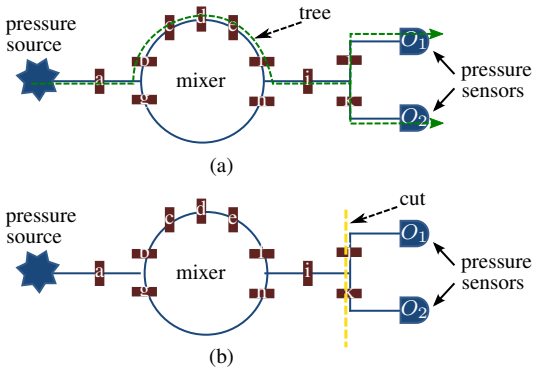


Fig. 16. A test tree and a cut on a traditional microfluidic biochip.

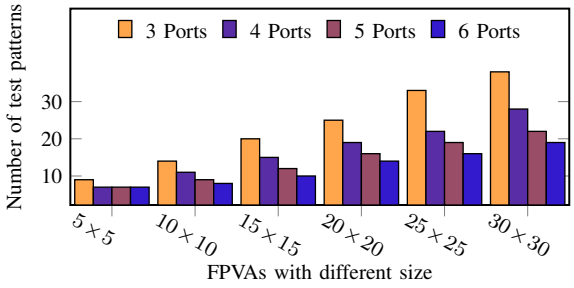


Fig. 17. Comparison of numbers of test patterns for FPVAs with different numbers of ports.

adapted to generate test patterns for traditional flow-based biochips.

Fig. 16(a) demonstrates a test tree on a traditional biochip. By opening all valves on the tree and closing the other valves, we can test the valves on the tree for stuck-at-0 faults. Fig. 16(b) shows a cut on this biochip. By closing all valves on the tree and opening the other valves, stuck-at-1 faults can be tested.

In generating test patterns for such a chip, its structure can be converted into a virtual valve array. The valves in the original chip stay unchanged, and any channel segment connecting valves can be viewed as a cell. Afterwards, the formulation described for FPVAs can be applied to identify the minimum set of test patterns. Since the test trees and cuts are the direct objectives of the proposed method, the number of test patterns can be lowered even further from the ATPG method in [19].

VI. SIMULATION RESULTS

The proposed framework was implemented in C++ and tested using a 3.20 GHz CPU with 8 GB memory. We demonstrate the results using FPVAs with valves in different rows and columns. The tested arrays are shown in Table I, where the first column shows the dimensions of the arrays and the second column shows the number of valves. We used Gurobi [32] to solve the optimization problems in the proposed method.

The results of testing FPVAs with a single pressure source and a single pressure sensor are shown in Table I. The direct approach is the method published in [1], which does not use loop relaxation introduced in Section IV-A4 for acceleration. In Table I, the numbers of test paths are shown in the column n_p^d. The CPU time to generate these patterns is shown in the column t_p^d. The columns n_c^d and t_c^d show the number of cut patterns and the CPU time to generate them. The columns n_l^d and t_l^d show the number of patterns for testing leakage in the control layer and the CPU time to generate these patterns. The total number of test patterns is shown in the column N^d. "*" means that no valid results are available due to runtime. According to these results, the direct approach suffers from the scalability problem due to its pure ILP formulation. In these results, the number of cuts is much larger than the number of test paths. This is because a path can travel through many valves in a zigzag shape and only a few orthogonal paths can already cover all the valves. However, the cuts required to cover all the valves in an FPVA is proportional to the size of the array, leading a much larger number of such test patterns.

The columns of loop acceleration show the results of generating test patterns using loop relaxation. The columns n_p^a, n_c^a and n_l^a show the numbers of test paths, cuts and leakage test patterns, respectively. The columns t_p^a, t_c^a and t_l^a show the CPU time needed to generate these patterns, respectively. The total number of the test patterns is shown in the column N^a. These results show that with loop acceleration valid test patterns can be generated for all these valve arrays, though the CPU time increases quickly as the size of the FPVA grows. For the case 30×30, the numbers of variables and constraints in the ILP formulation for generating the test patterns to detect stuck-at-0 faults, stuck-at-1 faults and control-layer leakage are already (17702,36910), (100920,1016247) and (20511,88552), respectively. Therefore, a long CPU time was required to solve the corresponding optimization problems.

The results of the proposed method for multiple-port test is shown in Fig. 17, where the numbers of total test patterns for FPVAs with different numbers of test ports are compared. With this comparison, it can be seen that the number of test patterns decreases quickly as the number of test ports increases. This is

TABLE II
COMPARISON BETWEEN TEST PATTERNS GENERATED BY THE PROPOSED METHOD AND THE ATPG-BASED METHOD

	Proposed method			ATPG-based method [19]	
	Test Pattern	Detected Faults		Test Pattern	Detected Faults
1	00100000111001111110000110000	17	1	11110101011000100101101010000	12
2	10101011101111110011000011101	5	2	01110010100101111010100010011	10
3	10111001111111100111111111111	2	3	11100111101001111001110011111	13
4	11111111011111111111111111111	1	4	11100100111011100011100100111	4
5	11011111111111111111111111111	1	5	01101001101011101000111101000	5
6	11111111110111111111101111111	1	6	11110110101001111101111111000	5
7	11111111111110111111111111111	1	7	00101010101101101011010110001	2
8	11111110111011111111111101111	24	8	10100111111101010100100111100	4
9	00100001101111110000000010111	11	9	00100110101101011101001000010	1
			10	00110100110111111010000000011	1
			11	01001010110101010100010011101	1
			12	10101001111000111000011010111	1
			13	10110100111101111110000100001	1
			14	11100010101101111010010011001	1
			15	10100110111111111101011000100	1
			16	10111101111011110100110011000	1

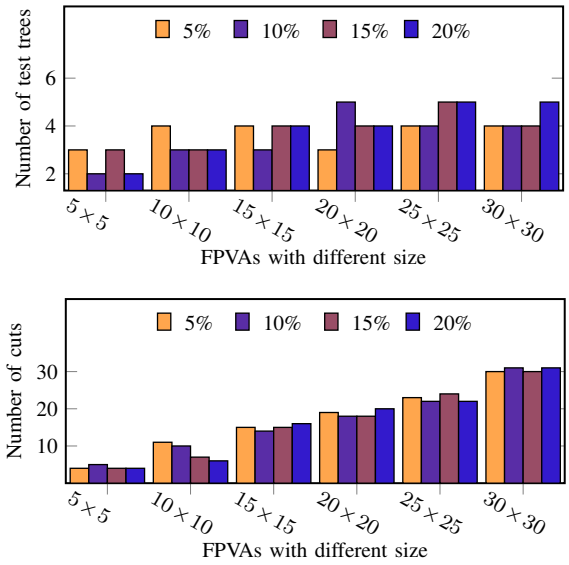


Fig. 18. Comparison of numbers of test patterns for FPVAs with different numbers of long channels and obstacles.

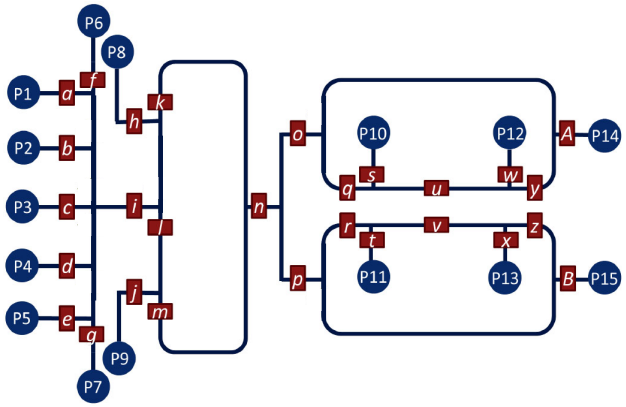


Fig. 19. A traditional multiple-port continuous-flow biochip from [19]. P1-P15 are ports. a-z, A and B are valves.

also valid when the number of test patterns with three ports is compared with that with two ports shown in Table I. Since the majority of the test patterns are cuts, more test ports enable a better coverage of valves by cuts composed of multiple segments, leading to a significant reduction of test patterns.

We also demonstrate the effectiveness of our method in dealing with long channels and obstacles in FPVAs. The results are shown in Fig. 18. In these cases, we randomly created FPVA layouts with 5%, 10%, 15% and 20% missing valves and obstacles. For each of these cases, one pressure source and two pressure sensors are assumed. As shown in this comparison, long channels and obstacles have only slight influence on the number of test patterns for FPVA test. However, it is worth noting that the real form of the test trees and cuts are completely different from those without long channels and obstacles.

We have applied the proposed method onto the traditional multiple-port continuous-flow biochip shown in Fig. 19 from [19]. The port $P3$ is connected to a pressure source. The ports $P1-P2$ and $P4-P15$ are connected to pressure sensors. Table II shows the comparison between the test patterns generated by our method and the test patterns generated by the ATPG method in [19] in detecting stuck-at-0 and stuck-at-1 faults. The test patterns define the open/closed states of valves in the order (a–z, A, B), and the numbers of individual faults that can be detected by the corresponding test patterns are also shown in the corresponding rows in Table II. The proposed method covers all stuck-at-0 and stuck-at-1 faults with 9 test patterns. With the method of converting the same biochip architecture into a circuit and then applying ATPG to generate test patterns as proposed in [19], 16 test patterns are needed. Therefore, the method proposed in this paper has a better efficiency in fault coverage.

To demonstrate the process of tree construction, the intermediate results of the 20×20 FPVA with long channels and obstacles are shown in Fig. 20. The FPVA has been abstracted into a graph, where nodes represent the cells and edges represent the valves. Randomly generated long channels and obstacles are created in the array. Fig. 20(a) shows the structure of the original FPVA, where the red edges represent always-closed valves or obstacles. The blue edges represent normal valves that are expected to be closed and opened during assay execution. Fig. 20(b) is the FPVA with its long channels

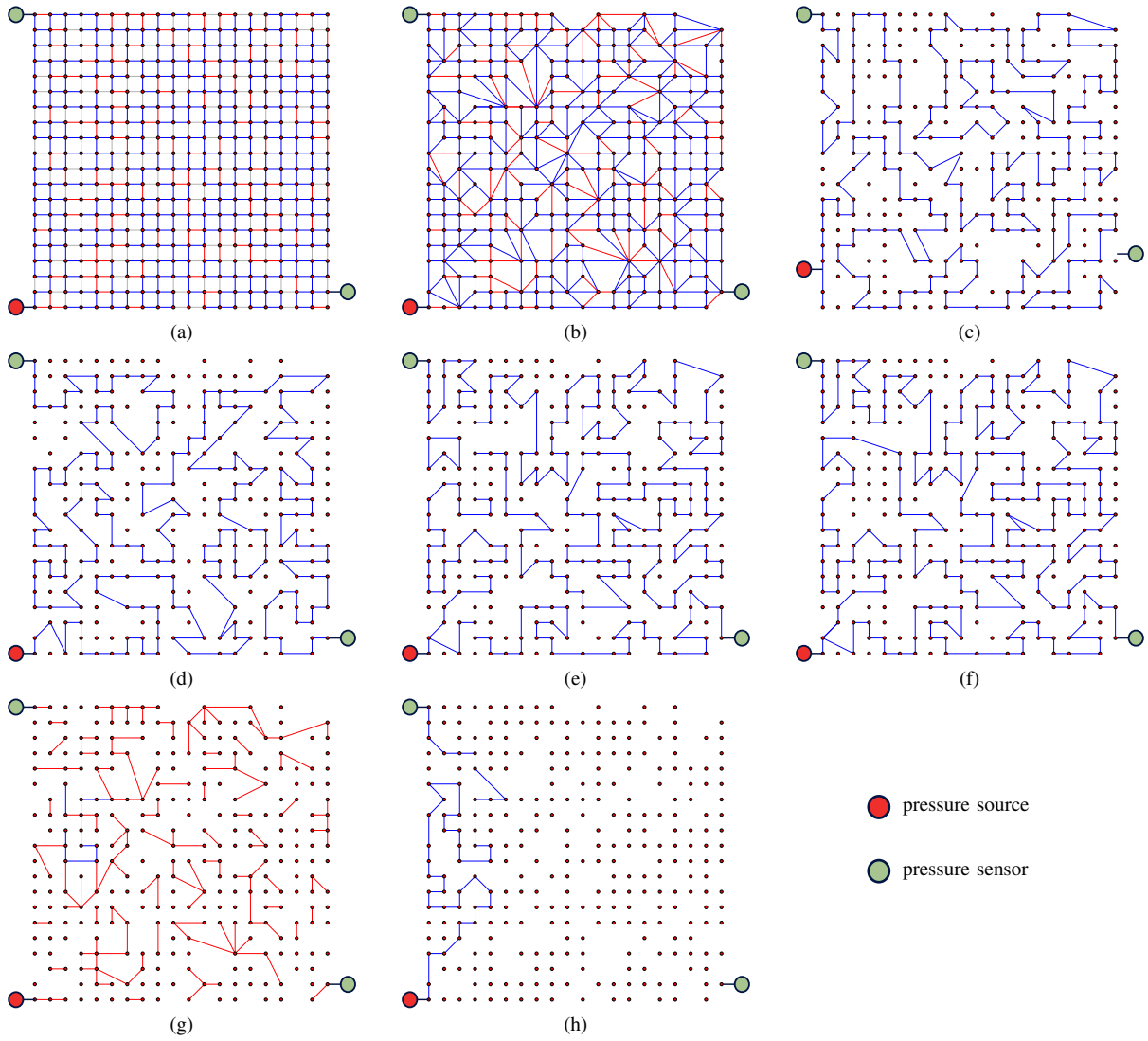


Fig. 20. Constructing test trees on a 20×20 FPVA with long channels and obstacles. (a) The original FPVA represented by a graph. (b) Long channels and obstacles are compressed into super cells. (c), (d) and (e) Three test trees with a loop in (e). (f) The previously constructed test tree in (e) is altered to partially cover the valves on the loop. (g) The remaining valves to cover. (h) One additional test tree to cover the remaining valves.

and obstacles merged as described in Section IV-D. Fig. 20(c), Fig. 20(d) and Fig. 20(e) are the test trees constructed using the ILP model with loop relaxation. There is a disjoint loop in Fig. 20(e) and it is altered in Fig. 20(f). Fig. 20(g) shows the remaining valves that need to be covered by additional trees after the loop is altered. In Fig. 20(h), one additional test tree is constructed to cover these valves.

To verify whether the test paths, cuts and the patterns for testing control layer leakage can detect faults reliably, for each FPVA in Table I we generated one, two, three, four and five faults separately and tested whether the test patterns can detect these faults. We repeated this process for 10 000 times. In these test cases, the test patterns detected all the faults, demonstrating a very high reliability in fault detection.

VII. Conclusion

In this paper, we have proposed the first strategy to generate test patterns with paths and cuts for fully programmable valve arrays (FPVAs) to detect faults such as blockage and leakage in the flow layer and the control layer. The method produces an efficient test set and can handle a vast majority of flow-based networks that can be configured on such an array. In the multiple-port test scenario, these patterns are expanded accordingly to take advantage of multiple pressure sensors to improve test efficiency. The proposed method can guarantee the detection of any two faults in FPVAs, while multiple faults can actually be captured effectively. When applied to traditional flow-based biochips, the generated patterns also demonstrate a high test efficiency. Identifying the location of faults in programmable array-based biochips may be investigated as a future research problem. Selective placement of test ports for further reduction of test sets is another open problem to study.

References

[1] C. Liu, B. Li, B. B. Bhattacharya, K. Chakrabarty, T.-Y. Ho, and U. Schlichtmann, "Testing microfluidic fully programmable valve arrays (FPVAs)," in *Proc. Design, Autom., and Test Europe Conf.*, 2017, pp. 91–96.

[2] J. Melin and S. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 213–231, 2007.
[3] J. M. Perkel, "Microfluidics: Bringing new things to life science," *Science*, vol. 322, no. 5903, pp. 975–977, 2008.
[4] K. Chakrabarty, "Design automation and test solutions for digital microfluidic biochips," *IEEE Trans. Circuits Syst.*, vol. 57, no. 1, pp. 4–17, 2010.
[5] [Online]. Available: http://www.illumina.com/
[6] Agilent. [Online]. Available: http://www.agilent.com/
[7] N. Amin, W. Thies, and S. P. Amarasinghe, "Computer-aided design for microfluidic chips based on multilayer soft lithography," in *Proc. Int. Conf. Comput. Des.*, 2009, pp. 2–9.
[8] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," in *Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embed. Sys.*, 2012, pp. 181–190.
[9] T.-M. Tseng, B. Li, U. Schlichtmann, and T.-Y. Ho, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 69–75, 2015.
[10] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, "Transport or store? Synthesizing flow-based microfluidic biochips using distributed channel storage," in *Proc. Design Autom. Conf.*, 2017, pp. 49:1–49:6.
[11] C. Lin, C. Liu, I. Chen, D. T. Lee, and T. Ho, "An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2014, pp. 141:1–141:6.
[12] K. Hu, T. Ho, and K. Chakrabarty, "Wash optimization and analysis for cross-contamination removal under physical constraints in flow-based microfluidic biochips," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 559–572, 2016.
[13] W. H. Minhass, P. Pop, J. Madsen, and T. Ho, "Control synthesis for the flow-based microfluidic large-scale integration biochips," in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2013, pp. 205–212.
[14] Q. Wang, S. Zuo, H. Yao, T.-Y. Ho, B. Li, U. Schlichtmann, and Y. Cai, "Hamming-distance-based valve-switching optimization for control-layer multiplexing in flow-based microfluidic biochips," in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2017, pp. 524–529.
[15] Y. Zhu, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips," in *Proc. Int. Conf. Comput.-Aided Des.*, 2018, pp. 123:1–123:8.
[16] K. Hu, T. A. Dinh, T. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 55–68, 2017.
[17] H. Yao, Q. Wang, Y. Ru, Y. Cai, and T. Ho, "Integrated flow-control codesign methodology for flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 60–68, 2015.
[18] H. Yao, T. Ho, and Y. Cai, "PACOR: practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2015, pp. 142:1–142:6.
[19] K. Hu, F. Yu, T. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1463–1475, 2014.
[20] K. Hu, T. Ho, and K. Chakrabarty, "Test generation and design-for-testability for flow-based mVLSI microfluidic biochips," in *Proc. VLSI Test Symp.*, 2014, pp. 1–6.
[21] C. Liu, B. Li, T.-Y. Ho, K. Chakrabarty, and U. Schlichtmann, "Design-for-testability for continuous-flow microfluidic biochips," in *Proc. Design Autom. Conf.*, 2018, pp. 164:1–164:6.
[22] A. Bernardini, C. Liu, B. Li, and U. Schlichtmann, "Fault localization in programmable microfluidic devices," in *Proc. Design, Autom., and Test Europe Conf.*, 2019.
[23] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab on a Chip*, vol. 11, pp. 1612–1619, 2011.
[24] ——, "Mixing operation in PMD," http://www.rsc.org/suppdata/lc/c0/c0lc00537a/videos3.mov, 2011.
[25] ——, "Fluid transportation in PMD," http://www.rsc.org/suppdata/lc/c0/c0lc00537a/videos2.mov, 2011.
[26] T. Tseng, B. Li, T. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, 2015, pp. 141:1–141:6.
[27] T.-M. Tseng, B. Li, M. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis with dynamic device mapping and fluid routing for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 12, pp. 1981–1994, 2016.
[28] G. Lai, C. Lin, and T. Ho, "Pump-aware flow routing algorithm for programmable microfluidic devices," in *Proc. Design, Autom., and Test Europe Conf.*, 2018, pp. 1405–1410.
[29] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2017, pp. 530–535.
[30] N. Madras and G. Slade, *The Self-Avoiding Walk*. Birkhäuser, 1996.
[31] D. Chen, R. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. Wiley, 2011.
[32] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2013. [Online]. Available: http://www.gurobi.com

Testing Microfluidic Fully Programmable Valve Arrays (FPVAs)

Chunfeng Liu[1,5], Bing Li[1], Bhargab B. Bhattacharya[2], Krishnendu Chakrabarty[3,5], Tsung-Yi Ho[4,5], Ulf Schlichtmann[1]
[1]Institute for Electronic Design Automation, Technical University of Munich, Germany [2]Indian Statistical Institute, Kolkata, India
[3]Department of ECE, Duke University, Durham, NC, USA [4]National Tsing Hua University, Hsinchu, Taiwan
[5]Institute for Advanced Study, Technical University of Munich, Germany
{chunfeng.liu, b.li, ulf.schlichtmann}@tum.de, bhargab@isical.ac.in, krish@ee.duke.edu, tyho@cs.nthu.edu.tw

***Abstract*—Fully Programmable Valve Array (FPVA) has emerged as a new architecture for the next-generation flow-based microfluidic biochips. This 2D-array consists of regularly-arranged valves, which can be dynamically configured by users to realize microfluidic devices of different shapes and sizes as well as interconnections. Additionally, the regularity of the underlying structure renders FPVAs easier to integrate on a tiny chip. However, these arrays may suffer from various manufacturing defects such as blockage and leakage in control and flow channels. Unfortunately, no efficient method is yet known for testing such a general-purpose architecture. In this paper, we present a novel formulation using the concept of flow paths and cut-sets, and describe an ILP-based hierarchical strategy for generating compact test sets that can detect multiple faults in FPVAs. Simulation results demonstrate the efficacy of the proposed method in detecting manufacturing faults with only a small number of test vectors.**

I. Introduction

Microfluidic biochips have revolutionized the traditional slow and error-prone biochemical experiment flow by manipulating nanoliter volumes of fluids precisely [1], [2], [3]. With this miniaturization, bioassays can be scaled down and genomic bioassay protocols, such as nucleic-acid isolation, DNA purification, and DNA sequencing, have been successfully demonstrated with these chips. In addition, this technology has also attracted a lot of commercial attention, e.g., from Illumina [4], a market leader in DNA sequencing.

Microfluidic biochips based on continuous flow use valves to control the movement of samples and reagents. The structure of a valve is shown in Fig. 1(a). In such a structure, a flow channel is constructed on a substrate for transportation of fluids. Above the flow channel, a control channel is constructed and connected to an air pressure source. Since both channels are built from elastic materials, air pressure applied in the control channel squeezes the flow channel tightly, so that the movement of the fluid is blocked. Conversely, if the pressure in the control channel is released, the fluid can resume its movement to the target destination. Consequently, a valve is formed at the intersection of the two channels.

Valves can also be used to build complex devices. For example, the structure of a mixer is shown in Fig. 1(b). When the three valves at the top of the mixer are actuated alternately by applying and releasing air pressure in the control channels, a circular flow around the device can be formed to mix different samples and reagents. After an operation is completed, the intermediate result can be transported to other devices or stored temporarily in a dedicated storage unit. Fig. 1(c) shows a detailed schematic of a mixer connected to a storage unit with eight cells. These neighboring storage cells can be constructed using normal flow channels and multiplexed control valves at both ends.

Recent advances in manufacturing technologies have enabled valve density to reach 1 million per cm^2 [6], and consequently,

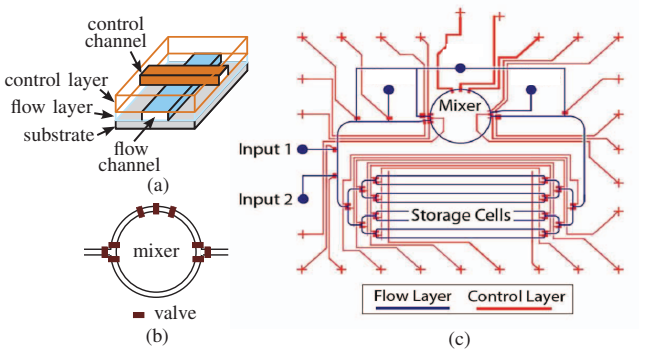


Fig. 1. Components and structure of flow-based biochips. (a) Valve structure. (b) Mixer. (c) Biochip with eight storage cells [5].

fully programmable valve arrays (FPVAs) have emerged for more flexible and highly reconfigurable flow-based biochips [1], [7]. Fig. 2(a) shows a part of the large valve array demonstrated in [7]. In this architecture, valves (solid blocks) are arranged in a regular manner along horizontal and vertical flow channels (light color). These valves are controlled by air pressure sources through the control channels (narrow channels). By opening two valves and closing the other two at a crosspoint of flow channels, the fluid sample stored there can be moved in the intended direction by forming temporary transportation channels.

Besides transportation channels, complex devices such as mixers can be constructed on the valve array by taking advantage of the flexibility and reconfigurability of such chips [8]. For example, a 4×2 mixer and a 2×4 mixer can be constructed as in Fig. 2(b) and 2(c), respectively. In such a dynamic mixer, the eight valves along the enclosed channel function as pump valves, which switch in a given pattern to drive the fluid samples and reagents inside the channel for mixing. Compared with the traditional mixer shown in Fig. 1(b), these dynamic mixers have a different shape and more pump valves, eight in each case, to form a strong circular mixing flow. The two mixers in Fig. 2(b) and 2(c) can share the same part of the chip area as shown in Fig. 2(d), provided that they are not used at the same time. Consequently, the same area of a valve array can execute various functions such as mixing and flow transportation, as well as detection if the corresponding sensors are included in this area.

It is convenient to fabricate FPVAs as large-scale integrated devices, because a regular structure is easy to design and manufacture compared with the traditional irregular fluidic architecture, similar to the case of DRAM-arrays in the semiconductor industry. In addition, dynamic reconfigurability enables the valve array to execute nearly any application. This flexibility allows chip vendors to focus on improving the integration scale without worrying about the applications. On the other hand, customers who use such chips also have the flexibility to perform different

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

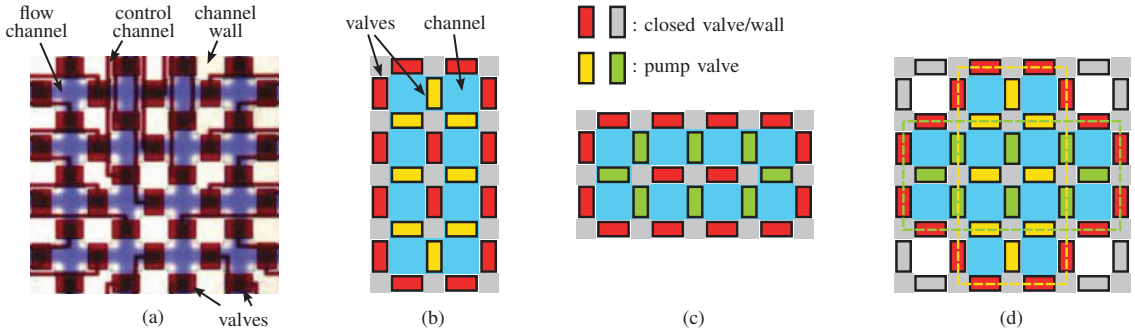


Fig. 2. Fully programmable valve array (FPVA). (a) Architecture [7]. (b)/(c) A 4×2/2×4 dynamic mixer. (d) Dynamic mixers of different orientations sharing the same area.

applications, a notable advantage specially for small healthcare centers or research laboratories that usually cannot afford expensive apparatus.

In order to facilitate industry adoption of FPVAs, efficient defect-screening techniques must be developed. In this paper, we focus on testing of FPVAs to identify chips with manufacturing defects. Our contributions are as follows:

- We propose the first systematic formulation and test strategy for detecting manufacturing defects efficiently in FPVAs;
- Our method is based on the construction of flow paths and cut-sets in the array, followed by a hierarchical ILP-based solution, which yields a very compact test set;
- The proposed method can guarantee the detection of up to two faults in the chip, while more than two faults can also be detected in nearly all cases;
- The method is general. It works both for a full array and an incomplete one with fluidic-seas (channels) or obstacles;
- The proposed test flow is compatible with test flows for traditional flow-based biochips, so that no additional cost is incurred for testing FPVAs.

The rest of this paper is organized as follows. In Section II, we review prior work on testing of traditional flow-based biochips and formulate the test problem for FPVAs. In Section III, we describe the general test strategy and explain how this strategy is implemented. Simulation results are reported in Section IV. Conclusions are stated in Section V.

II. Fault Model and Problem Formulation

During the manufacturing of FPVAs, various defects may occur, as illustrated in Fig. 3. These defects have been analyzed in detail and the corresponding fault models have been defined in [9]. Based on how these defects affect the behavior of a valve or a channel, faults at the component level can be defined as follows:

- *A break in a flow channel*: Fluid cannot pass through a channel. This is equivalent to the fault that the valve at the entrance of the channel cannot be opened.
- *A break in a control channel*: Air pressure cannot reach a valve to close it.
- *Leaking flow channel*: Fluid in a channel leaks to neighboring channel. In FPVA test, this fault is similar to the fault that a valve cannot be closed, because there is always a valve between two channels.
- *Leaking control channel*: Two valves close simultaneously due to the shared pressure in the control layer.

In the following discussion, if a valve cannot be opened, we refer to the scenario as a *stuck-at-0 fault*. Similarly, if a valve cannot be closed, we refer to the scenario as a *stuck-at-1 fault*.

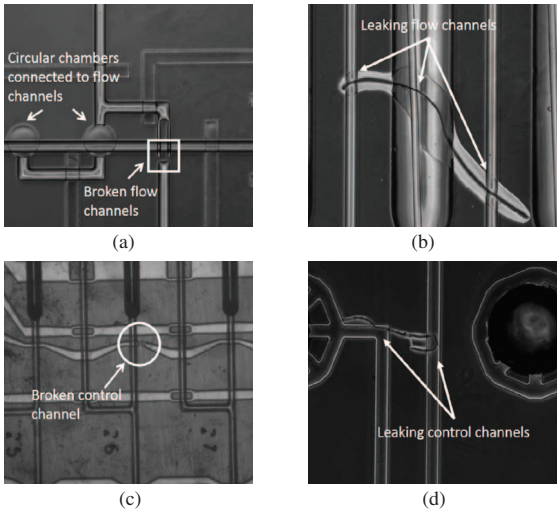


Fig. 3. Manufacturing defects in flow-based biochips [9]: (a) Broken flow channel; (b) Leaking flow channel; (c) Broken control channel; (d) Leaking control channel.

To detect faults in a flow-based biochip, the method in [9] connects air pressure sources to the input ports of the biochip to generate a pressure in the flow layer. By switching some valves open or closed with different control input vectors, air pressure patterns at the output ports can be used to deduce whether there is a fault in the chip. For example, in Fig. 4, a pressure can be detected at the output o_2 if the valves a, g, h, i, k are open, while the other valves are closed. However, if there is a valve on this path that cannot be opened, there is no pressure at o_2, indicating the existence of a stuck-at-0 fault. On the other hand, if a valve on this path, e.g., g, is closed, while the other valves are open, there is no path from the pressure source to the output ports. If a pressure can still be detected at the output port, at least one stuck-at-1 fault exists.

In the above analysis, it can be seen that test results at the output ports carry fault information of internal valves, similar to the testing of integrated circuits. Therefore, the method in [9] represents the relationship between paths and valves using a digital circuit model and generates ATPG vectors to identify faults. This method offers the advantage of adopting a mature

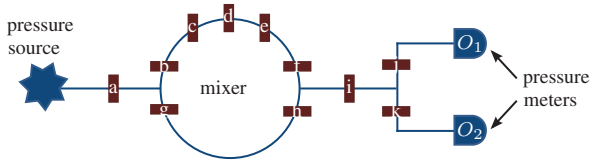


Fig. 4. Example of a biochip under test.

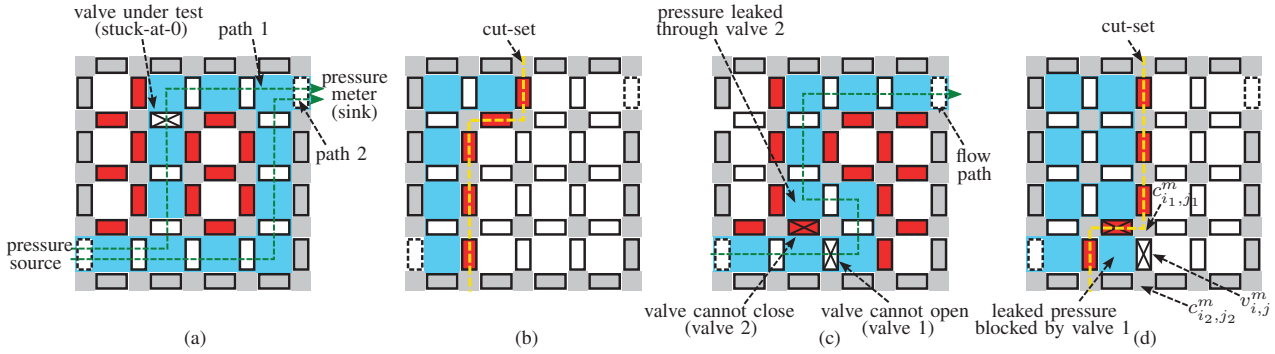


Fig. 5. Flow paths and cut-sets. Valves at the external boundary of the chip are always closed. (a) Flow paths masking a stuck-at-0 fault. (b) Cut-set. (c) Flow path with two-fault masking. (d) Cut-set with two-fault masking.

test flow by mapping the test problem to an ATPG problem. It, however, faces the challenge of generating a very large circuit model or many circuit models for testing an FPVA, because valves in an FPVA can be reconfigured to form a huge number of dynamic chip architectures, all of which should be covered by the ATPG-based method.

In this paper, we propose a new test framework that can detect faults in a manufactured chip reliably with only a small set of test vectors. This problem can be formulated as follows:

Inputs: An FPVA architecture; the locations of valves that are not built on flow channels (conceptually always open), and the locations of obstacles (conceptually always closed); the locations of the air pressure sources and the pressure meters.

Outputs: A set of test vectors, where each vector defines the open/closed states of all valves when test pressure is applied at the sources and checked at the output ports by the pressure meters.

Objectives: The number of test vectors to detect faults in an FPVA should be as small as possible to reduce test cost. The number of undetected faults should be as small as possible.

III. FPVA TEST WITH FLOW PATHS AND CUT-SETS

In this section we explain the strategy to test an FPVA and the implementation of the test method. Although omitted from this paper due to the space limit, leakage at the control layer can also be detected by adapting the valve coverage problem described in this section. For convenience, we refer to a pressure source simply as a source port, and to a pressure meter port as a sink port in the following discussion.

A. Test strategy for an FPVA

To identify whether a fault exists in a chip, the test vectors should ensure that an error is observable when valves are switched. For example, if no test vector opens a valve during the test process, the only fault that can be observed at this valve is the leakage fault (stuck-at-1 fault) and the fault that the valve cannot be opened (stuck-at-0 fault) is not tested. Therefore, the test vectors should switch a valve open at least once and closed at least once during test application to detect stuck-at-0 and stuck-at-1 faults at this valve. For each case, the effect of the correct behavior of the valve should be observable at sink ports. For example, if a valve is switched open, there should be at least one path from the source through this valve to the sink to detect the pressure being transmitted through this valve.

When test vectors are applied, some faults might mask each other. For example, if another path that circumvents the valve under test connects the pressure source and the sink port, the potential stuck-at-0 fault (always closed) at the valve under test may be masked and thus cannot be observed. This situation is illustrated in Fig. 5(a). In this example, the two paths are created on the chip at the same time, and thus the stuck-at-0 fault at the valve under test cannot be detected because there is still a pressure at the sink port due to the second path. To avoid this path interference problem, we only construct simple paths without loops or branches. These paths are called *flow paths* henceforth.

Similar to constructing flow paths to detect stuck-at-0 faults, we construct cut-sets to detect stuck-at-1 faults. A *cut-set* is formed by a set of valves that separate the source ports and the sink ports completely. In test application, if all the valves in a cut-set are closed and a pressure is still detected by a pressure meter, a stuck-at-1 fault must exist. An example of such a cut-set is illustrated in Fig. 5(b), which disconnects any path between the source and the sink.

In a scenario with multiple faults, the flow-path and cut-set vectors discussed above, however, cannot guarantee that a fault can always be detected. Assume that there are two faulty valves, one of which cannot be opened (valve 1, stuck-at-0) and the other cannot be closed (valve 2, stuck-at-1). Also assume that the flow path used to test valve 1 is constructed as shown in Fig. 5(c), and the cut-set used to test valve 2 is constructed as shown in Fig. 5(d). In Fig. 5(c), the pressure leakage through valve 2 masks the stuck-at-0 fault at valve 1. In Fig. 5(d), the pressure leakage through valve 2 is blocked by valve 1. In both cases, the results at the pressure meter are still correct, so that these two faults cannot be detected. Consequently, mutual masking patterns of this type should be excluded from the generated test vectors.

B. Generating flow-path test vectors

In the proposed method, we generate the flow paths using an Integer Linear Programming (ILP) model. The scalability of this model is improved using a hierarchical approach.

1) Constructing flow paths

A fluid cell is defined as the channel area surrounded by four valves, as shown in Fig. 6(a). Air pressure through a cell must pass through two of the valves surrounding this cell. Since the air pressure can reach the cell in any direction, in total there are 12 possible directions for a path passing through this cell as shown in Fig. 6(a). Instead of modeling theses directions directly, we model how the path passes through the surrounding valves. Suppose all valves can be covered by no more than n_p flow paths in the test set, where n_p is a given constant. For the cell at the location of the ith row and the jth column of the valve array, we assign a 0-1 variable $c^m_{i,j}$ to represent whether the mth path

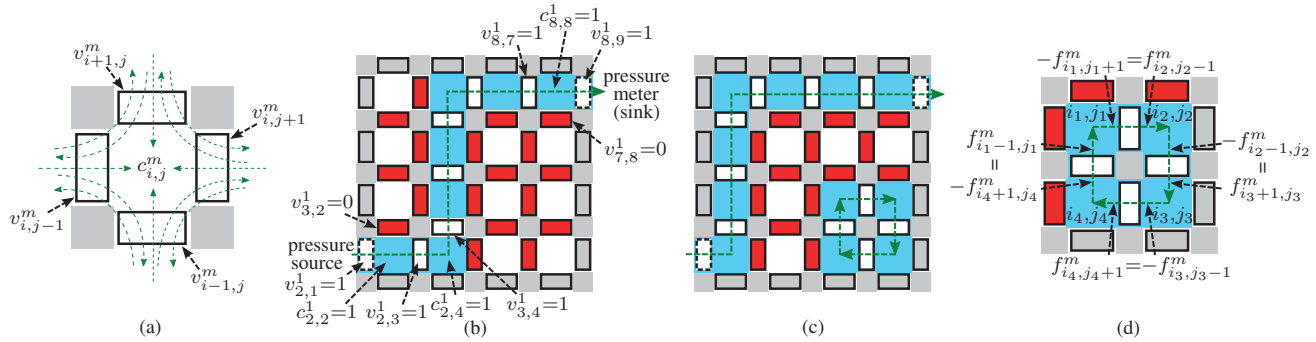


Fig. 6. Flow path model. (a) Constraint variables for valves and cells. (b) Path construction using constraints. (c) Disjoint loop. (d) Flow constraints along a disjoint loop.

passes through the cell. If the mth path passes through the cell, $c^m_{i,j}$=1; otherwise $c^m_{i,j}$=0 . For the valves at the left, right, upper and lower sides of the cell at the location (i,j), we assign 0-1 variables $v^m_{i,j-1}$, $v^m_{i,j+1}$, $v^m_{i+1,j}$ and, $v^m_{i-1,j}$, respectively. If the mth path passes through a valve, the corresponding variable is set to 1; otherwise, it is set to 0.

If the mth path passes through the cell at the location (i,j), this path should pass through exactly two valves that surround the cell; otherwise, no valve surrounding the cell should be passed. Consequently, the relation between the cell and the valves surrounding it can be established as

$$v^m_{i,j-1}+v^m_{i,j+1}+v^m_{i+1,j}+v^m_{i-1,j}=2c^m_{i,j},\ \forall\ i=1,2,...,n_r,\ j=1,2,...,n_c,\ \ m=1,2,...,n_p \quad (1)$$

where n_r and n_c are numbers of rows and columns of the valve array, respectively. Constraint (1) constructs the mth path by the chaining effect of the variables $v^m_{i,j}$ as demonstrated in Fig. 6(b).

To guarantee that a valve is covered at least once by the flow paths, one of the constraint variables $v^m_{i,j}$ for the valve indexed by (i,j) on the m paths must be one, leading to

$$\sum_{m=1}^{n_p} v^m_{i,j}\geq 1,\ \forall\ i=1,2,...,n_r,\ \ j=1,2,...,n_c. \quad (2)$$

2) Excluding disjoint flow loops

With the constraints (1) and (2), disjoint loops may appear on the flow paths. For example, the constraint does not prevent the disjoint loop at the lower right side of the valve array in Fig. 6(c) from happening. All the valves and cells on the loop meet the constraints (1) and (2), but this loop gives a false counting of valve coverage in testing, because pressure from the source cannot reach a valve on this loop so that it is not possible to test whether the valves on the loop can be opened.

To solve the disjoint loop problem, we force the air pressure from the source to reach any segment of the path. To represent the pressure volume (pressure flow) passing through a valve at the location (i,j), we define an integer variable $f^m_{i,j}$. This variable is positive when viewed from a cell which the pressure flow enters; it is negative when viewed from a cell which the pressure flow leaves. In addition, a pressure can pass through a valve only if the valve is on the test path, under the condition $v^m_{i,j}$=1. Otherwise $f^m_{i,j}$ should be set to 0. This condition constrains $f^m_{i,j}$ as

$$f^m_{i,j}\leq v^m_{i,j}\cdot\mathcal{M} \quad \text{and} \quad f^m_{i,j}\geq -v^m_{i,j}\cdot\mathcal{M} \quad (3)$$

where $\mathcal{M}$ is a large positive constant [10]. Consequently, the pressure volume in the cell at (i,j) when testing the mth flow path is equal to the sum of the volumes of the four surrounding valves. This relation can be written as

$$f^m_{i,j-1}+f^m_{i,j+1}+f^m_{i+1,j}+f^m_{i-1,j}=c^m_{i,j} \quad (4)$$

where the cells on the left of, on the right of, above and below the cell at the location (i,j) are indexed by $(i,j-1)$, $(i,j+1)$, $(i+1,j)$ and $(i-1,j)$, respectively.

Constraint (4) prevents disjoint loops from appearing effectively. Assume there is a disjoint loop on the mth path and the cells on the disjoint loop are indexed by (i_1,j_1) to (i_l,j_l), where the valves at (i_l,j_l) and (i_1,j_1) are neighbors. For each cell on the loop we can write a constraint similar to (4). Adding the left and right sides of these constraints together, we have

$$\sum_{(i,j)\in I_l}(f^m_{i,j-1}+f^m_{i,j+1}+f^m_{i+1,j}+f^m_{i-1,j})=\sum_{(i,j)\in I_l} c^m_{i,j} \quad (5)$$

where I_l is the index set $\{(i_1,j_1)...(i_l,j_l)\}$ for the cells on the loop. On a disjoint loop, the sum on the left side of (5) is always equal to 0, because no pressure flow enters the loop. This contradicts the fact that $\sum_{(i,j)\in I_l} c^m_{i,j}$ should be larger than 0, because the cells are on the flow path. The concept of this model is illustrated in Fig. 6(d).

3) Finding the minimum set of flow paths

The path constraints defined above rely on a known number n_p paths that can guarantee the coverage of all valves. In our formulation, we first assign n_p a constant and then try to find a set of paths whose number is no larger than n_p that can cover all valves. For each path, we assign a 0-1 variable p_m,$1\leq m\leq n_p$ to indicate whether this path is used. Because any valve on the mth path marks the path to be used, p_m can be constrained as

$$p_m\cdot\mathcal{M}\geq \sum_{(i,j)\in I} v^m_{i,j} \quad (6)$$

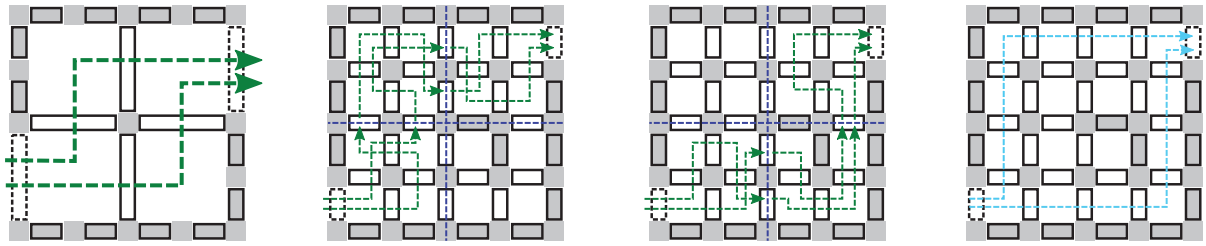
where $\mathcal{M}$ is a positive constant larger than the number of valves on the array, and I is the index set of all valves. If a valve is on the mth path, the right side of (6) is larger than 0, so that p_m must be set to 1 to meet the constraint.

With the constraints above, the ILP problem to find a minimum set of paths that cover all valves can be formulated as follows,

$$\text{minimize} \quad \sum_{m=1}^{n_p} p_m \quad (7)$$

$$\text{subject to} \quad (1)-(4) \text{ and } (6). \quad (8)$$

Since we specify the number of paths n_p as a constant, it is possible that the ILP problem above has no solution, meaning that not all the valves can be covered by n_p flow paths. If this happens, we increase n_p and solve the optimization problem again.



(a) (b) (c) (d)

Fig. 7. Hierarchical path construction and valve search for cut-sets. (a) Paths at the top level indicate flow direction. (b)/(c) Subpaths in subblocks forming final flow paths. (d) Searching beginning and ending valves of cut-sets. Vales along the two search directions form the starting and ending valve sets.

4) Improving scalability with a hierarchical model

To improve the scalability of the proposed method, we apply a hierarchical approach, where we partition the valve array into subblocks and find the test paths in each subblock individually. Afterwards, the flows paths at the top level are formed by connecting the subpaths at lower levels. With this technique, the proposed method can process large designs more efficiently but generates more test vectors.

The concept of this hierarchical model can be explained using the example in Fig. 7(a)-(c), where the original FPVA is a 4×4 array, and we consider each 2×2 array as a subblock. Consequently, the top level becomes a 2×2 array. Thereafter, we apply the ILP formulation described above to find the top level flow paths, as shown in Fig. 7(a). The flow paths at the top level define the direction of the subpaths in the subblocks. For example, if there is a path from the top left to the top right and the right boundary of the subblock has two valves, there should be at least two paths in the flow direction in the subblocks to cover those two valves. The test paths in each subblock are generated by solving the ILP problem similar to (7)–(8), while constraining further that these subpaths start and end at the boundary of the subblock at the side where the top-level paths enter and leave the subblock. Finally, the final test paths are created by connecting the subpaths in different subblocks, as shown in Fig. 7(b)-(c). The rule of this connection is that a subpath should be included at least once, so that the valves it covers are covered by the final test paths.

C. Generating cut-set test vectors

Besides flow paths, we also need to create cut-sets to test whether all valves can be closed. Because a cut-set separates the source and the sink, an end of a cut-set must touch an edge of the chip, as shown in Fig. 5(b) and 5(d). Observing this phenomenon, we search the valves along the boundary of the chip in two directions starting from the source until the sink port is reached from both directions, as shown in Fig. 7(d). Consequently, we find two sets of valves, and a cut-set must include a valve from each of these sets.

The cut-set generation problem is a complementary problem of finding a set of flow paths covering all valves described in Section III-B and can be solved by adapting the optimization problem (7)–(8) to include the additional constraint that a cut-set must start and end at the boundary of the chip.

As discussed in Section III-A, we must prevent the pattern shown in Fig. 5(c) and 5(d) from appearing in the cut-set to guarantee the detection of two faults masking each other. This illegal pattern can be described as a new cut-set can be formed by only one new valve with some valves from the old cut-set. Assume there is a valve at the location (i,j) and the two obstacle areas at the two ends of the valve are indexed by (i_1,j_1) and (i_2,j_2). To prevent this pattern from being formed, we add an additional constraint to the optimization problem as

$$c^m_{i_1,j_1}+c^m_{i_2,j_2}-1\leq v^m_{i,j} \tag{9}$$

which specifies that if the two ends of a valve are in the current cut-set, this valve must be included in the cut-set to prevent the illegal pattern in Fig. 5(c) and 5(d).

IV. Simulation Results

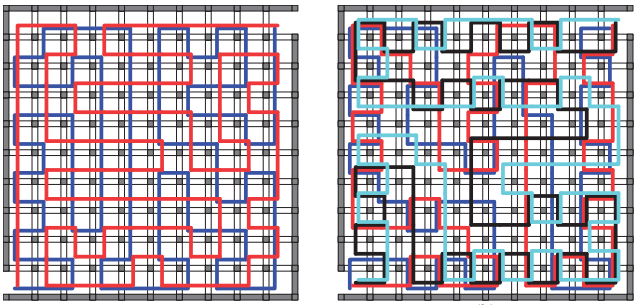
The proposed framework was implemented in C++ and tested using a 3.20 GHz CPU with 8 GB memory. We demonstrate the results using FPVAs of different rows and columns. The tested arrays are shown in Table I, where the first column shows the dimensions of the arrays and the second column shows the number of valves. These arrays contain long channels for transportation and obstacle areas without valves.

In the simulation, the dimension of subblocks was set to 5×5 and the arrays were partitioned regularly at the top level. The hierarchy of the test cases is shown in the third and the fourth columns in Table I. The numbers of test vectors generated for flow paths and cut-sets are shown in the columns n_p and n_c, respectively. Although not explained in detail in this paper, the proposed method can also be adapted to generate test vectors to detect control layer leakage. The numbers of these test vectors are shown in the column n_l in Table I. The total number of test vectors are shown in the column N. These numbers are roughly two times of the square root of the numbers of valves in the arrays. Since this is the first method proposed for FPVA fault test, we do not have a way to compare the efficiency of the proposed method. However, consider a simple baseline method where only one valve is switched open or closed each time for fault test. The total number of test vectors in this case would be two times of the number of valves, a squared complexity compared with the proposed method.

The runtimes for generating each set of test vectors are shown in the columns t_p, t_c, and t_l, respectively. The total runtime for generating all these vectors is shown in the column T. For generating the flow-path and cut-set vectors, the proposed model was solved in a few minutes. For generating the vectors to test control layer leakage, the largest array used about 25 minutes. The large runtime results from the facts that the problem to find a minimal set of paths in an undirected graph to cover all nodes is NP-hard, and to guarantee the detection of any two faults by excluding the patterns shown in Fig. 5(c) increases the complexity of the problem tremendously. In practice, the proposed method needs to be executed offline only once to generate the test vectors for a given array architecture, so that the runtime is already acceptable. However, the improvement of the efficiency of the model is still our focus, and we are currently

TABLE I
RESULTS OF TEST VECTOR GENERATION

Valve Array		Hierarchy		Flow Paths		Cut-sets		Control Leakage		Total	
Dimension	n_v	Top	Subblock	n_p	$t_p(s)$	n_c	$t_c(s)$	n_l	$t_l(s)$	N	$T(s)$
5 × 5	39	1 × 1	5 × 5	5	0.3	8	0.2	4	2	17	2.5
10 × 10	176	2 × 2	5 × 5	4	4	18	5	4	10	26	19
15 × 15	411	3 × 3	5 × 5	8	17	28	26	8	127	44	170
20 × 20	744	4 × 4	5 × 5	16	35	38	41	16	742	70	818
30 × 30	1704	6 × 6	5 × 5	20	255	58	171	20	1492	98	1918



(a) (b)

Fig. 8. Comparison of the hierarchical model to the direct model for a 10×10 FPVA. (a) Two flow paths from the direct ILP model. (b) Four paths from the hierarchical model.

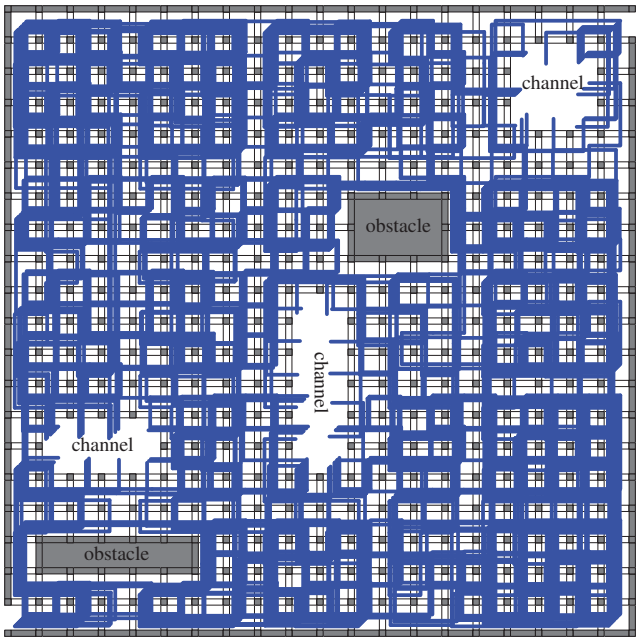


Fig. 9. A total of 16 flow paths for the 20×20 array with channels and obstacles.

testing a vector-based path generation model which can eliminate many variables so that even much larger valve arrays can be processed in a reasonable time.

To demonstrate the effectiveness of the hierarchical model, we show the flow paths of a 10×10 valve array without channels or obstacles. In Fig. 8(a), the flow paths are generated using the ILP model without hierarchy directly. It is interesting to observe that only two flow paths are needed to cover all the valves in this regular array. In Fig. 8(b) the flow paths for the same array are generated using a hierarchy with the subblock dimension 5×5. The number of paths in this case is four, a little larger than the number from the direct model, but still acceptable.

To demonstrate the generated test vectors, we show the flow paths for the 20×20 valve array from Table I in Fig. 9. With only 16 paths, all the 744 valves in this array are covered. In this array, there are three channels and two obstacles, demonstrating that the proposed method can also deal with FPVAs with irregular structures efficiently.

To verify whether the combination of flow paths and cut-sets can detect faults as explained in Section III, for each valve array in Table I we randomly introduced one, two, three, four and five faults, respectively, and applied the generated test vectors. We repeated this process 10 000 times. In these test cases, the test vectors captured all the faults. Therefore, we can conclude that these test vectors are very effective in practice in detecting faults, although in theory they can only guarantee the detection of two faults in a chip.

V. CONCLUSION

We have proposed the first strategy to detect manufacturing faults in fully programmable valve arrays (FPVAs). We have also introduced a hierarchical ILP-based model to identify a small set of test vectors. The proposed method can guarantee the detection of any two faults in a chip, and it also demonstrated its potential in detecting more than two faults in a chip in our simulation.

ACKNOWLEDGMENT

The work of B. Li and U. Schlichtmann was supported by the IGSSE Project FLUIDA of Technical University of Munich. The work of B. B. Bhattacharya was supported, in part, by the special research grant funded by PPEC, Indian Statistical Institute, Kolkata. The work of Chunfeng Liu was supported fully, and the work of K. Chakrabarty and T.-Y. Ho was supported in part, by the Technical University of Munich – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763. The work of T.-Y. Ho was also supported in part by the Ministry of Science and Technology of Taiwan, under Grant MOST 105-2221-E-007-118-MY3.

REFERENCES

[1] J. Melin and S. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 213–231, 2007.
[2] J. M. Perkel, "Microfluidics: Bringing new things to life science," *Science*, vol. 322, no. 5903, pp. 975–977, 2008.
[3] K. Chakrabarty, "Design automation and test solutions for digital microfluidic biochips," *IEEE Trans. Circuits Syst.*, vol. 57, no. 1, pp. 4–17, 2010.
[4] [Online]. Available: http://www.illumina.com/
[5] N. Amin, W. Thies, and S. P. Amarasinghe, "Computer-aided design for microfluidic chips based on multilayer soft lithography," in *Proc. Int. Conf. Comput. Des.*, 2009, pp. 2–9.
[6] I. E. Araci and S. R. Quake, "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves," *Lab Chip*, vol. 12, pp. 2803–2806, 2012.
[7] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab Chip*, vol. 11, pp. 1612–1619, 2011.
[8] T. Tseng, B. Li, T. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, 2015, pp. 141:1–141:6.
[9] K. Hu, F. Yu, T. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1463–1475, 2014.
[10] D. Chen, R. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. Wiley, 2011.