Built-In Self-Test for Locating Leakage Defects on Continuous-Flow Microfluidic Chips

Jiahui Peng, Mengchu Li, Tsun-Ming Tseng, Ulf Schlichtmann

Technical University of Munich

{jiahui.peng, mengchu.li, tsun-ming.tseng, ulf.schlichtmann}@tum.de

***Abstract*—Continuous-flow microfluidic chips (CFMBs) are effective platforms for bio-experiments using small volumes of fluids. Inside these chips, fluids are transported through flow channels and controlled by valves, which are actuated by pressure applied via control channels. As the scale of CFMBs grows, the likelihood of blockage and leakage defects increases, highlighting the growing importance of thorough chip testing. However, current leakage testing methods focus on detecting the existence of defects but cannot precisely locate the defective channels. This paper proposes the first coding methodology for the design of a built-in self-test (BIST) module that can precisely locate one or multiple leakage defects, generally with an extra cost of no more than two flow channels. Based on the new BIST module, a novel locating method is then proposed to reduce the number of test operations. For example, given 128 control channels and compared to the state-of-the-art approaches, our method reduces the average number of test operations by up to 85.9% and 76.6%, given one or two pair(s) of random leaky control channels, respectively.**

I. Introduction

Continuous-flow microfluidic biochips (CFMBs) are promising platforms for bio-experiments [1]–[4]. A CFMB consists of a flow layer with flow channels for fluids and a control layer with control channels for pressure. Membrane valves, shared by both layers, regulate the fluid behavior. When a valve is pressurized by the control channel connected to it, it closes the fluid pathways in the corresponding flow channel. Taking advantage of the precise control over small volumes of fluids, large bio-experiments can be implemented on tiny chips, increasing the automation degree and decreasing the material costs [5]. However, the miniature components, particularly control channels, suffer a risk of blockage and leakage defects [6], [7]. Defective control channels cannot pressurize valves as intended to govern the fluid behavior, affecting the functions of the chip and resulting in erroneous experimental results. Therefore, testing the chips before use is essential.

State-of-the-art testing methods for CFMBs include functional test (FCT) [8], [9] and built-in self-test (BIST) [10], [11]. FCT methods generate the test patterns with automatic test pattern generation (ATPG) tools, pressurize the flow channels, measure the pressure states with sensors, and compare the measurement with the expected results to conclude the existence of defects. However, since the control channels are tested indirectly via flow channels, some defects may be masked. BIST methods integrate extra test modules onto the chip, extend the control channels from the on-chip devices to the test modules, and carry out the test operations in the test modules. At the cost of moderate resource overhead, BIST methods enable better fault coverage with fewer test operations and no extra equipment/software costs. [10] (referred to as BIST1) proposes the first BIST module that detects control channel defects at arbitrary positions. However, the blockage and leakage defects may mask each other when they happen in the same channels. [11] (referred to as BIST2) proposes separate blockage and leakage test modules to prevent blockage and leakage defects from masking each other. In particular, the new blockage module tests any number of control channels with constantly one operation, and the new leakage module greatly simplifies the tests by taking advantage of the fact that leakage only happens between nearby parallel channels [12].

Compared to blockage which affects only a single channel, leakage involves multiple channels, making it more difficult to test. Current leakage test methods generally assume that chips should be discarded once any defect is identified, and thus only focus on detecting the existence of defects but not locating the defects. Moreover, with the presence of multiple defects, current methods cannot guarantee to detect all of them. However, since a CMFB usually consists of many channels, the chip may still be usable even if a few channels are defective. By locating all defects, one can modify the operational protocol to minimize or even eliminate the impact of the defects [13]. Besides, identifying the exact locations of defective channels enables designers to adjust the chip layout, addressing frequently failing channels and improving overall yield.

In this work, we design the first BIST module that precisely locates one or multiple leakage defects, based on Sperner's theory and a vertex coloring model. With our new design, we further propose a method that combines a vertex cover model with a bisection-based algorithm to reduce the number of test operations for locating the defects. Simulation results show that compared to the state-of-the-art, our BIST module generally requires an extra cost of no more than two flow channels, even for large-scale CMFBs with thousands of control channels, and our test method greatly improves the efficiency of the tests.

II. Background

A. Control channel defects

Common control channel defects include blockage and leakage. Blockage means that a control channel is broken and

thus the pressure transportation path is blocked, making some valves on the blocked channel unable to be closed. Leakage means that two control channels are connected unintentionally, causing pressure to leak from one channel to the other. As a result, and the valves on the leaky channels are opened or closed together. Fig. 1 illustrates how the fluid behavior deviates from expectation when defects occur. Control and flow channels are colored in green and blue, respectively, and valves are represented by green rectangles.

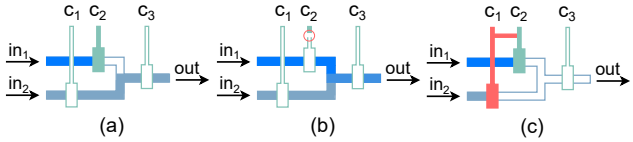

Fig. 1. (a) The desired operation is to collect fluid at *out* from in_2. (b) A blockage on c_2 results in contamination at *out* by fluids from in_1. (c) A leakage between c_1 and c_2 leads to transportation failure.

B. BIST for CFMBs

BIST methods divide a CFMB into a main circuit and a test module (combining a blockage module and a leakage module), as shown in Fig. 2. The control channels of the test module are extended from the main circuit, and the flow channels of the test module are independent of the main circuit, ensuring that any introduced test fluids, such as food dye, do not contaminate the main circuit. A test operation injects fluids from the left port of the test module and inspects the fluid status near the right port of the test module. With specific test patterns for pressurizing the control channels, defective control channels in the main circuit will result in faulty behavior of valves in the test module, and thus faulty fluid status at the inspection area.

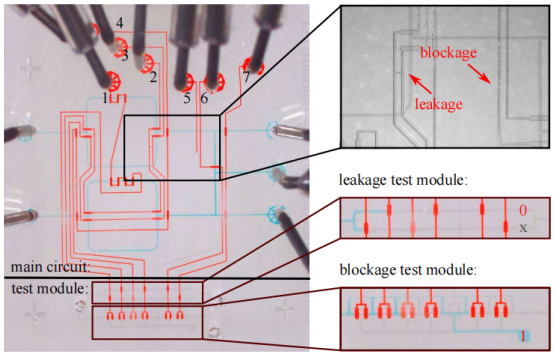

Fig. 2. An mLSI chip integrated with the test module [11]. The control layer and flow layer of the main circuit are colored in red and blue, respectively.

The state-of-the-art blockage test [11] can detect and locate the blockage defects of any number of control channels with constantly one test operation, while the state-of-the-art leakage test cannot guarantee to detect and locate all leakage defects.

C. Basic leakage test

We refer to the state-of-the-art leakage test [11] as a *basic leakage test*. Given a set of parallel and neighboring control channel pairs that have the risk of leakage, the basic leakage test decides whether the leakage defect exists or not.

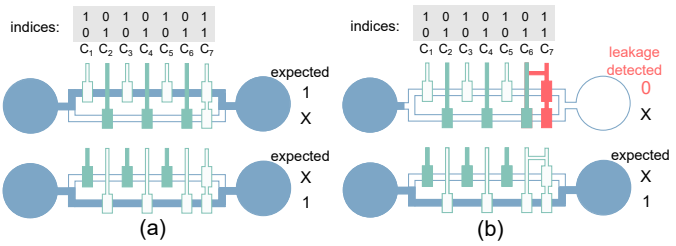

Fig. 3. (a) Expected results of a basic leakage test. (b) When leakage happens between $c_6(01) \in \bar{C}_{f_1}$ and $c_7(11) \in C_{f_1}$, the test operation on the 1st flow channel outputs a faulty result, showing that the indices of the leaky channels differ in the 1st bit.

For a leakage module consisting of n flow channels, every control channel connected to the module is assigned with an n-bit binary code as its index, representing the presence of valves on the corresponding flow channels. For example, Fig. 3 shows a leakage module consisting of 2 flow channels. Each control channel c connected to the module is thus assigned with a 2-bit index, where the 1^{st} bit represents whether c has a valve on the 1^{st} flow channel, denoted as f_1, and the 2^{nd} bit represents whether c has a valve on the 2^{nd} flow channel, denoted as f_2.

The number of test operations in a basic leakage test is equal to the number of flow channels. Each test operation selects a flow channel f_i, and pressurizes all control channels that do not have valves on f_i (i.e. with '0' in the i-th bits of their indices), denoted as a set $\bar{C}_{f_i}$, while depressurizes all control channels that have valves on f_i (i.e. with '1' in the i-th bits of their indices), denoted as a set C_{f_i}, such that if there is no leakage between a channel c in $\bar{C}_{f_i}$ and a channel $c' \in C_{f_i}$, fluids should flow from the left port through f_i to the right port, as shown in Fig. 3(a); but if there is a leakage between a channel $c \in \bar{C}_{f_i}$ and a channel $c' \in C_{f_i}$, c' will be unintentionally pressurized and thus block f_i, preventing fluids from reaching the right port, as shown in Fig. 3(b). By ensuring that the indices of any two control channels c and c' that have the risk of leakage differ by at least one bit, it is ensured that there is at least one flow channel f_i such that $c \in \bar{C}_{f_i}$ and $c' \in C_{f_i}$.

D. Shortcomings of the basic leakage test

The basic leakage test can only identify the bits in which the indices of the leaky control channels differ. To locate the defects, BIST1 [10] proposed to individually pressurize every control channel to find out the indices of the leaky channels. Not to mention that pressurizing every channel results in a large number of test operations, but it also cannot guarantee to locate the defects. First, since channels that do not have the risk of leakage may share the same indices, identifying the indices doesn't necessarily lead to the identification of leaky channels. Second, even if every control channel is assigned with a unique index, pressurizing a leaky channel does not necessarily lead to faulty fluid behavior. For example, as shown in Fig. 4(a), despite the leakage between c_6 and c_7, pressurizing c_7 results in fluid behavior that matches the expectation. This is because c_7 has a valve on every flow channel on which c_6

has a valve. In other words, pressurizing c_7 blocks all flow channels that c_6 can block, and thus whether c_6 is pressurized or not makes no difference to the fluid behavior in the test module. Under the assumption that there is only one leakage defect, it is still possible to derive that c_7 is defective based on the result of the basic leakage test. However, when there are multiple leakage defects, some defects may be masked and no longer derivable. For example, when c_5 leaks with both c_6 and c_7, both test operations in the basic leakage test output faulty results, showing that the indices of the leaky channels differ in the first two bits, as shown in Fig. 4(b). However, by pressurizing individual control channels, only c_5 and c_6 output faulty results, while c_7 shows a result as expected, as shown in Fig. 4(c). Since c_5 (10) and c_6 (01) differ in the first two bits, it is natural to draw the wrong conclusion that they are the only leaky channels, leaving the defect in c_7 masked.

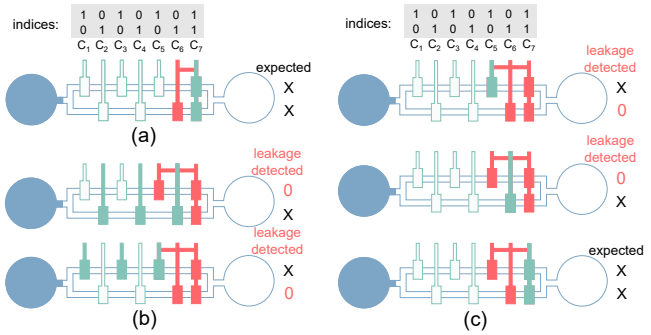

Fig. 4. (a) Pressurizing c_7 shows a result as expected. (b) Results of the basic test operation in case c_5 leaks with both c_6 and c_7. (c) Pressurizing c_5 and c_6 shows faulty results, but pressurizing c_7 shows a result as expected.

III. Design Principles for Locating All Leakage-defective Control Channels

To prevent multiple leakage defects from being masked by one another, we need to make sure that for every pair of control channels c and c' that have the risk of leakage, c has at least one valve in a flow channel where c' does not have a valve, and c' has at least one valve in a flow channel when c does not have a valve. In this manner, pressurizing either c or c' will lead to faulty fluid behavior that deviates from the expectation, and thus, the channel defect can be located.

For example, as shown in Figure 4(c), c_5 has a valve in the 1st flow channel where c_6 does not have a valve, and c_6 has a valve in the 2nd flow channel where c_5 does not have a valve. Thus, when c_5 leaks with c_6, pressurizing c_5 will additionally block the 2nd flow channel, which deviates from the expectation, and pressurizing c_6 will additionally block the 1st flow channel, which deviates from the expectation, too. Thus, both defective control channels can be identified directly without referring to the result of the basic leakage test.

To formally state the above description as a design principle for a BIST module to locate all control channel leakage defects, we denote the binary code index of every control channel c with two sets: $\mathbb{1}_c$ and $\mathbb{0}_c$, which contain the positions of all '1'- and '0'-bits, respectively. For example, the binary code index of control channel c_6 in Fig. 4 is 01, i.e. the 2nd bit of the index is '1', while the 1st bit of the index is '0'. Thus, $\mathbb{1}_{c_6} = \{2\}$ and $\mathbb{0}_{c_6} = \{1\}$. We then state the design principle as follows:

For any two control channels c and c' that have the risk of leakage, $\mathbb{1}_c$ and $\mathbb{1}_{c'}$ must not be a strict subset of each other.

A module with N_f flow channels can encode a control channel index with N_f bits. To encode as many control channels as possible with the fewest flow channels while satisfying the above-mentioned design principle, we need to find the largest set E of subsets of $\{1, \cdots, N_\mathrm{f}\}$, in which each subset is not a strict subset of another. According to Sperner's theorem [14]:

$$|E| \leq C_{N_\mathrm{f}}^{\lceil \frac{N_\mathrm{f}}{2} \rceil} = C_{N_\mathrm{f}}^{\lfloor \frac{N_\mathrm{f}}{2} \rfloor}, \tag{1}$$

in which $C_{N_\mathrm{f}}^{\lceil \frac{N_\mathrm{f}}{2} \rceil}$ is calculating the number of different combinations to choose $\lceil \frac{N_f}{2} \rceil$ items from a total of N_f items. The equality holds if and only if E consists of all subsets that have size $\lceil \frac{N_\mathrm{f}}{2} \rceil$ or $\lfloor \frac{N_\mathrm{f}}{2} \rfloor$. In other words, if the size of $\mathbb{1}_c$, denoted as $|\mathbb{1}_c|$, for every control channel c is $\lceil \frac{N_f}{2} \rceil$ or $\lfloor \frac{N_f}{2} \rfloor$, we can encode the most control channels with the fewest flow channels. It is worth mentioning that the size of $\mathbb{1}_c$ is not strictly required to be $\lceil \frac{N_\mathrm{f}}{2} \rceil$ or $\lfloor \frac{N_\mathrm{f}}{2} \rfloor$, if the number of to-be-indexed control channels is smaller than $C_{N_\mathrm{f}}^{\lceil \frac{N_\mathrm{f}}{2} \rceil}$. For example, given 4 control channels, at least 4 flow channels are required, as $6 = C_4^2 \geq 4 > C_3^1 = 3$. Nevertheless, $|\mathbb{1}_c|$ is not restricted to 2 for every control channel c. It can also be set to 1 or 3, since $C_{N_f}^{|\mathbb{1}_c|}$ still satisfies the requirement: $C_4^1(4) = C_4^3(4) \geq 4$.

IV. Algorithm for Designing Leakage Test Module

To design a test module that locates all leakage defects, we need to assign an index to every control channel such that the indices of control channels that have the risk of leakage satisfy the design principle discussed in Section III. To improve the resource and test efficiency of the module, we aim to minimize the number of flow channels in the module, denoted as N_f. Under the prerequisite of minimizing N_f, we also minimize $|\mathbb{1}_c|$ for every control channel c, since the leakage information is only revealed by pressurizing the '0'-bits. Moreover, fewer '1'-bits indicate fewer valves and thus a more reliable design [15].

Algorithm 1 shows our design method. Specifically, given the to-be-tested control channels as input, we first construct a graph model $G = (V, E)$ in which a vertex $c \in V$ represents a control channel, and an edge $(c_i, c_j) \in E$ is added between every pair of control channels $c_i, c_j \in V$ that have the risk of leakage, as shown in line 1. The target of the algorithm is to decide the binary code index of every control channel $c \in V$, as shown in line 2. To this end, we first perform vertex coloring on G with DStatur algorithm. The color of a vertex c, denoted as r_c, represents the binary code index of c. Thus, by coloring incident vertices differently, control channels that have the risk of leakage are indexed differently. Based on the coloring solution, we contruct a set R for all colors, and thus the size of R, denoted as $|R|$ represents the minimum number of different binary codes required to index the control channels, as shown in line 3. To minimize N_f, we

Algorithm 1 Design control channel indices

1: **Input:** a graph $G = (V, E)$ in which each vertex $c \in V$ represents a control channel, and each edge $(c_i, c_j) \in E$ represents that there is risk of leakage between c_i and c_j.
2: **Output:** a binary code index for every $c \in V$
3: Minimize the number of different indices [16] by solving the vertex coloring problem on G using the DSatur algorithm. Denote the color of each vertex c as $r_c \in R$, and denote the number of required colors as $|R|$.
4: $N_f \leftarrow \lceil \log_2(|R| + 1) \rceil$
5: **while** $C_{N_f}^{\lfloor N_f/2 \rfloor} < |R|$ **do**
6: $N_f \leftarrow N_f + 1$
7: **end while**
8: $|\mathbb{1}_c| \leftarrow \lfloor N_f/2 \rfloor$
9: **while** $|R| \leq C_{N_f}^{|\mathbb{1}_c|-1}$ **do**
10: $|\mathbb{1}_c| \leftarrow |\mathbb{1}_c| - 1$
11: **end while**
12: Enumerate all binary codes with N_f bits and $|\mathbb{1}_c|$ '1'-bits, and map each code to a color index in R.

initialize N_f as $\lceil \log_2(|R| + 1) \rceil$, i.e., the smallest number of bits to represent $|R|$ different binary codes. To ensure that the indices satisfy the design principle discussed in Section III, we increase N_f until $C_{N_f}^{\lfloor N_f/2 \rfloor} >= |R|$, as shown in lines 4–7. Next, to minimize $|\mathbb{1}_c|$, we initialize $|\mathbb{1}_c|$ as $\lfloor N_f/2 \rfloor$, and decrease $|\mathbb{1}_c|$ if $C_{N_f}^{|\mathbb{1}_c|-1} >= |R|$, i.e., if the number of different combinations of the binary codes with $|\mathbb{1}_c| - 1$ '1'-bits is larger than or equal to the required number of control channel indices, as shown in lines 8–11. At last, we decide the index of each control channel by enumerating all binary codes with N_f bits and $|\mathbb{1}_c|$ '1'-bits and map each code to a color $r_c \in R$ as the index of control channel c. Figure 5 illustrates an example of designing a leakage module applying Algorithm 1.

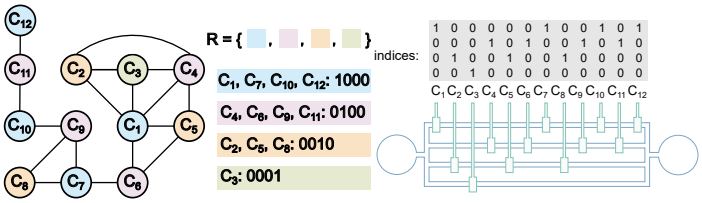

Fig. 5. Based on the vertex coloring solution of an input graph G with $|R| = 4$, we initialize N_f as $\lceil \log_2(4 + 1) \rceil = 3$ and update it to 4 because $C_3^{\lfloor 3/2 \rfloor} = 3 < 4$, and we initialize $|\mathbb{1}_c|$ as $\lfloor 4/2 \rfloor = 2$ and update it to 1 because $C_4^1 >= 4$. We then enumerate all binary codes with 4 bits and 1 '1'-bits to decide the index of each control channel and obtain the leakage test module.

V. Method for Locating Leakage Defects

With our new leakage test module, individually pressurizing a leaky control channel will definitely lead to a faulty test result that cannot be masked. However, when there are many to-be-tested control channels, pressurizing them one by one, as proposed in BIST1 [10], is time-consuming. Thus, we proposed a new test method to reduce the number of test operations for locating the defects, as described in Algorithm 2.

Algorithm 2 Algorithm of locating leaky control channels

1: **Input:** the graph $G = (V, E)$ as in Algorithm 1, the binary code index of every control channel $c \in V$.
2: **Output:** a set D containing all leaky control channels
3: Perform the basic leakage test.
4: **for** $(c_i, c_j) \in E$ **do**
5: **if** c_i does not leak with c_j according to the basic leakage test results **then**
6: Remove (c_i, c_j) from E
7: **end if**
8: **end for**
9: **for** $c \in V$ **do**
10: **if** c has degree 0 **then**
11: Remove c from V
12: **end if**
13: **end for**
14: Find the minimum vertex cover $K \subseteq V$ of G with MILP.
15: Divide K into disjoint subsets $K_1, \cdots, K_n \subseteq K$, where each subset consists of all control channels with the same binary code index. Denote the set of subsets as S'.
16: **for** $K_i \in S'$ **do**
17: Delete K_i from S'
18: Pressurize all control channels in K_i
19: **if** the test result is faulty **then**
20: **if** $|K_i| > 1$ **then**
21: Divide K_i into two disjoint subsets $K_{i_1}, K_{i_2} \subseteq K_i$ with equal number of elements and add K_{i_1} and K_{i_2} to S'
22: **else if** $|K_i| = 1$ **then**
23: Add the only control channel in K_i into D
24: **end if**
25: **end if**
26: **end for**
27: **for** $c_i \in V$ and $c_i \notin K$ **do**
28: **if** $\exists c_j \in D$ such that (c_i, c_j) satisfies the test result **then**
29: Pressurize c_i
30: **if** the test result is faulty **then**
31: Add c_i into D
32: **end if**
33: **end if**
34: **end for**

Specifically, we first perform the basic leakage test, as shown in line 3. Based on the test results, we identify control channel pairs that do not leak with others, and remove the corresponding edges from the input graph G. After that, we remove all vertices with degree 0 from G, because a vertex without incident edges represents a control channel without leakage, and thus does not need to be tested for locating the leakage. The updated process is as shown in lines 4–13. To further minimize the number of control channels that we need to individually pressurize, we find the minimum vertex cover K on G with the following mixed integer linear programming (MILP) model:

$$\forall (c_i, c_j) \in E, \quad b_{c_i} + b_{c_j} \geq 1, \tag{2}$$

$$\text{Minimize: } \sum_{c \in V} b_c, \tag{3}$$

where b_c is a binary variable declared for every $c \in V$ representing whether c is in the vertex cover. By minimizing the smallest number of vertices that cover all edges, we minimize the number of to-be-pressurized control channels that cover all leakage possibilities, as shown in line 14. To identify leaky channels with the same binary code indices, we divide K into disjoint subsets, each of which can be pressurized as a whole, and perform a bisection method to iteratively divide the subsets until the leaky channel is identified, as shown in lines 15–26. In this way, the number of control channels that need to be individually pressurized is further reduced. At last, the remaining control channels $c \in V \setminus K$ are tested individually, if c has an edge with any faulty control channel in D and meanwhile satisfies the test result, as shown in lines 27–34. Figure 6 illustrates Algorithm 2 using the same case as Figure 5.

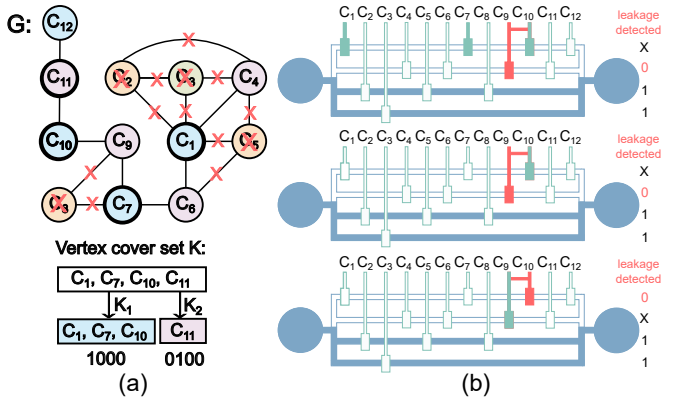

Fig. 6. (a) Suppose that the basic leakage test outputs faulty results only on the 1^{st} and 2^{nd} flow channels, we can conclude that control channels with indices differ in the 3rd or the 4th bits are free from leakage, and thus update G accordingly. We then find the minimum vertex cover $K = \{C_1, C_7, C_{10}, C_{11}\}$ of G and divide K into two subsets: $K_1 = \{c_1, c_7, c_{10}\}$ contains control channels with index 1000 and $K_2 = \{c_{11}\}$ contains control channels with index 0100. (b) The faulty test results during the locating process are shown. Suppose that pressurizing all channels in K_1 shows a faulty result, while pressurizing K_2 does not, we conclude that c_1, c_7, or c_{10} may be defective. K_1 is further divided into two subsets $\{c_1, c_7\}$ and $\{c_{10}\}$. Suppose that pressurizing c_{10} shows a faulty result and the faulty result outputs on the 2^{nd} flow channel, while pressurizing c_1 and c_7 together shows the result as expected, we conclude that c_{10} is leaky. At last, c_9 is tested, because it is not in the vertex cover set, but is connected to c_{10} in the graph and satisfies the test result of c_{10}. The result shows that c_9 is also defective.

VI. Results

A. Resource efficiency of the leakage test module

We evaluate the resource efficiency of our leakage test module based on seven chips from published literature, as shown in Table I. Each chip is modeled as an input graph G, as discussed in Section IV. Besides the number of control channels, i.e., the number of vertices in G, denoted as $|V|$, and the number of pairs of control channels that have the risk of leakage, i.e., the number of edges in G, denoted as $|E|$, we also show two other features of the inputs: the average and the maximal numbers of control channels to which a control channel leaks, i.e., the average and the maximal degrees of vertices in G, denoted as $\bar{\sigma}$ and $\sigma_{\max}$, because the degrees of the vertices usually affect the vertex coloring solutions, i.e., the smallest number of different control channel indices, denoted as $|R|$, and the number of valves on a control channel c in our leakage module, denoted as $|\mathbb{1}_c|$. We notice that although the chips have significantly different input features, $|R|$ remains ≤ 6, which matches the results reported in BIST2 [11], since the chromatic number of the vertex coloring problem generally does not increase with the scale of the graph. As a result, the number of flow channels in the test module, denoted as N_f, stays 4 for all test cases. The value of $|\mathbb{1}_c|$ varies between 1 and 2, reducing unnecessary valve usage while minimizing N_f.

TABLE I
Resource usage of our leakage test module

Case	$\lvert V\rvert$	$\lvert E\rvert$	$\bar{\sigma}$	$\sigma_{\max}$	$\lvert R\rvert$	N_{f}	$\lvert\mathbb{1}_c\rvert$
nucleic acid [17]	14	20	3	4	4	4	1
WGA [18]	23	49	4	11	6	4	2
cell-free bio. net. [19]	24	48	4	10	4	4	1
kinase activity [20]	24	64	5	8	5	4	2
mRNA [21]	25	57	5	10	5	4	2
ChIP (syn.) [22]	31	51	3	6	4	4	1
ChIP [23]	46	240	5	11	5	4	2

$|V|$: number of control channels; $|E|$: number of control channel pairs that have the risk of leakage; $\bar{\sigma}$: average number of the control channels to which a control channel leaks; $\sigma_{\max}$: the maximum number of control channels to which a control channel leaks; $|R|$: number of different binary coding indices of the control channels; N_{f}: number of flow channels in our leakage module; $|\mathbb{1}_c|$: number of valves on each control channel in our leakage module.

To explore the extreme situation, we also synthesize our leakage test module on synthesized test cases with $|R|$ up to 100. Figure 7 shows the comparison between our design, denoted as BIST3, with the state-of-the-art BIST2 design, which can only perform the basic leakage test but cannot detect and locate all leakage defects. As shown in the figure, compared to BIST2, even in the extreme cases with very large $|R|$, our design only uses 1 or 2 additional channels, which can be considered a small and reasonable sacrifice for precisely detecting and locating all defects. We observe that the number of additional flow channels decreases from 2 to 1 in some cases. This is because when $|R|$ increases, the combinatorial expression required by our design does not have the same increasing rate as the logarithmic expression required by BIST2.

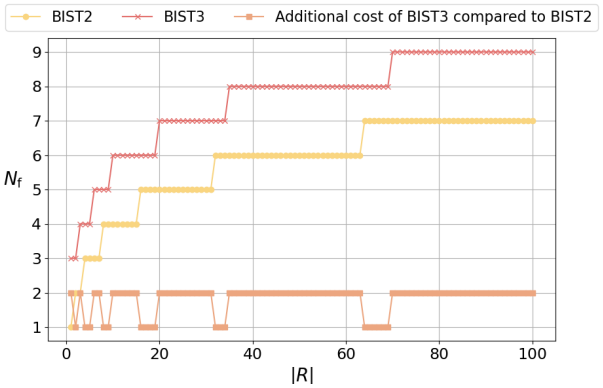

Fig. 7. Comparison of N_{f} between BIST2 and BIST3.

B. *Efficiency of the test method for locating the defects*

We evaluate the efficiency of our test method for locating defects with synthesized test cases. Given a number of control channels and a $\sigma_{\max}$, i.e., the largest number of control channels, with which a control channel has the risk of leakage, we exhaustively explore all the leakage possibilities between any channel c_i and any other channel in the range of $[i-\sigma_{\max}, i+\sigma_{\max}]$. We compared our method with the state-of-the-art defect locating method proposed in BIST1 [10], which pressurizes all control channels one by one, referred to as the baseline method.

To begin with, we enumerate all cases with one random leakage pair and set $\sigma_{\max}=6$. Fig. 8 illustrates the correlation between the number of control channels, denoted as $|V|$, and the number of test operations for locating the leaky channel pair, denoted as N_{test}. As the number of control channels increases from 16 to 2048, N_{test} in our method, denoted as BIST3, increases from 9 to 30. We performed a linear fit for $\log_2(|V|)$ and the N_{test} in BIST3. The correlation coefficient is 0.9962, indicating a strong linear relationship. However, the N_{test} in the baseline method increases linearly with $|V|$. When $|V|=16$, i.e., the number of control channels is 16, BIST3 saves 43.8% of the test operations. When $|V|$ increases to 2048, 98.5% of the test operations are saved. In general, with one random leakage pair, the advantages of our method become more pronounced as the number of control channels increases.

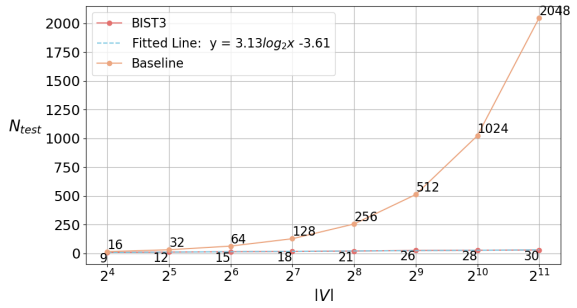

Fig. 8. Comparison of N_{test} between BIST3 and the baseline method, supposing one random leaky control channel pair.

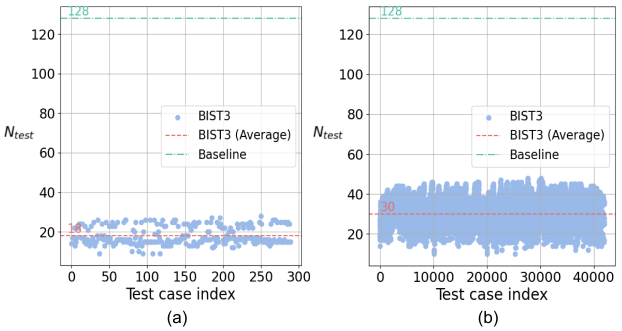

Fig. 9. Comparison of N_{test} between BIST3 and baseline methods with 128 control channels and 290 random relevant pairs under: (a) one random leakage channel pair; (b) two random leakage channel pairs.

We then enumerate cases with two random leaky pairs with 64 and 128 control channels and set $\sigma_{\max}=6$. Fig. 9 illustrates the distribution of N_{test} for all test cases with one or two leaky pairs among 128 control channels. We notice that when the number of leaky pairs increases from one to two, N_{test} in BIST3 rises from 15 to 24 and 18 to 30 for 64 and 128 control channels, respectively. Thus, to locate two leaky pairs, the number of test operations is less than twice that of locating only one leaky pair. For 128 control channels, 85.9% and 76.6% reductions in test operations are achieved with one or two pairs of randomly leaky control channels, respectively. As shown in Fig. 9, even the value of our worst case is much better than the baseline value. However, as the number of leaky control channel pairs increases, the advantage of BIST3 gradually diminishes. But with a large number of defects, the location of defects becomes less important, because the large number of faulty control channels indicates that the manufacturing process may have serious flaws, and the chip is unlikely to be reusable. In general, the more control channels there are and the fewer leaky pairs exist, the better performance BIST3 has.

We also explore the influence of $\sigma_{\max}$ on N_{test} with cases consisting of 128 control channels. In practice, $\sigma_{\max}$ is unlikely to exceed 12, as shown in Table I. We thus explore $\sigma_{\max}$ in the range from 6 to 18. Table II presents the relationship between $\sigma_{\max}$ and N_{test} in our method, showing that increasing $\sigma_{\max}$ leads to only a moderate rise in test operations. Even with $\sigma_{\max}=18$, our method still achieves an 82.8% reduction in test operations compared to the baseline, which requires 128 test operations. It is noteworthy N_{test} does not always grow monotonically, because $\sigma_{\max}$ also affects $|\mathbb{1}_c|$, i.e., number of '1'-bits in the binary coding indices, which further affects N_{test} as discussed in Section IV.

TABLE II
IMPACT OF $\sigma_{\max}$ ON THE NUMBER OF TEST OPERATIONS

$\sigma_{\max}$	6	7	8	9	10	11	12	13	14	15	16	17	18
N_{test}	18	18	20	20	20	21	21	21	21	22	21	22	22

$\sigma_{\max}$: maximum number of control channels to which a control channel may leak; N_{test}: number of test operations to locate all leaky control channels.

Finally, we want to report on the efficiency of our MILP models for the vertex cover problems. We implement the models in Python and solve them using Gurobi [24] on an Intel Core i9-13900HX processor. Although solving MILP models is NP-hard, our models for all test cases are solved within 0.1 seconds. This is because by minimizing $|\mathbb{1}_c|$ and updating graph G, as discussed in Algorithm 2, we effectively reduce the problem size of the vertex cover problem, making it possible for MILP to find the optimal solutions within a very short time.

VII. CONCLUSION

The reliability of continuous-flow microfluidic chips is crucial and has attracted increasing attention over the past decades. This paper introduces the first BIST leakage module design that can precisely detect and locate any number of defects, with an additional cost of no more than two flow channels compared to the state-of-the-art design. Based on the new leakage module, we propose an efficient test method for locating the leakage with a significantly reduced number of test operations.

ACKNOWLEDGMENT

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 515003344.

REFERENCES

[1] Jing Wu, Ziyi He, Qiushui Chen, and Jin-Ming Lin. Biochemical analysis on microfluidic chips. *TrAC Trends in Analytical Chemistry*, 80:213–231, 2016.

[2] Huanhuan Shi, Kaixuan Nie, Bo Dong, Mengqiu Long, Hui Xu, and Zhengchun Liu. Recent progress of microfluidic reactors for biomedical applications. *Chemical Engineering Journal*, 361:635–650, 2019.

[3] Prapti Pattanayak, Sachin Kumar Singh, Monica Gulati, Sukriti Vishwas, Bhupinder Kapoor, Dinesh Kumar Chellappan, Krishnan Anand, Gaurav Gupta, Niraj Kumar Jha, Piyush Kumar Gupta, et al. Microfluidic chips: recent advances, critical strategies in design, applications and future perspectives. *Microfluidics and nanofluidics*, 25:1–28, 2021.

[4] Farnaz Dabbagh Moghaddam and Francesca Romana Bertani. Application of microfluidic platforms in cancer therapy. *Materials Chemistry Horizons*, 1(1):69–88, 2022.

[5] Genggeng Liu, Hongbin Huang, Zhisheng Chen, Hongxing Lin, Hui Liu, Xing Huang, and Wenzhong Guo. Design automation for continuous-flow microfluidic biochips: A comprehensive review. *Integration*, 82:48–66, 2022.

[6] Ying Zhu, Xing Huang, Bing Li, Tsung-Yi Ho, Qin Wang, Hailong Yao, Robert Wille, and Ulf Schlichtmann. Multicontrol: Advanced control-logic synthesis for flow-based microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2489–2502, 2019.

[7] Siyuan Liang, Mengchu Li, Tsun-Ming Tseng, Ulf Schlichtamnn, and Tsung-Yi Ho. Combinatorial-coding-based high-performance microfluidic control multiplexer: Design, synthesis, and adaptation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[8] Kai Hu, Tsung-Yi Ho, and Krishnendu Chakrabarty. Testing of flow-based microfluidic biochips. In *2013 IEEE 31st VLSI Test Symposium (VTS)*, pages 1–6. IEEE, 2013.

[9] Kai Hu, Feiqiao Yu, Tsung-Yi Ho, and Krishnendu Chakrabarty. Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1463–1475, 2014.

[10] Mengchu Li, Yushen Zhang, Ju Young Lee, Hudson Gasvoda, Ismail Emre Araci, Tsun-Ming Tseng, and Ulf Schlichtmann. Integrated test module design for microfluidic large-scale integration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6):1939–1950, 2022.

[11] Mengchu Li, Hanchen Gu, Yushen Zhang, Siyuan Liang, Hudson Gasvoda, Rana Altay, Ismail Emre Araci, Tsun-Ming Tsneg, Tsung-Yi Ho, and Ulf Schlichtmann. Late breaking results: Efficient built-in self-test for microfluidic large-scale integration (mlsi). In *ACM/IEEE Design Automation Conference (DAC)*, 2024.

[12] Jiaxing Wang, Mingxin Zheng, Wei Wang, and Zhihong Li. Optimal protocol for moulding pdms with a pdms master. *Chips & Tips (Lab on a Chip)*, 6, 2010.

[13] Mengchu Li, Jiahui Peng, Tsun-Ming Tseng, and Ulf Schlichtmann. Ft-mux: A fault-tolerant microfluidic multiplexer design. In *ACM/IEEE Design Automation Conference (DAC)*, 2025.

[14] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

[15] Siyuan Liang, Meng Lian, Mengchu Li, Tsun-Ming Tseng, Ulf Schlichtmann, and Tsung-Yi Ho. Armm: Adaptive reliability quantification model of microfluidic designs and its graph-transformer-based implementation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.

[16] Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[17] Jong Wook Hong, Vincent Studer, Giao Hang, W French Anderson, and Stephen R Quake. A nanoliter-scale nucleic acid processor with parallel architecture. *Nature biotechnology*, 22(4):435–439, 2004.

[18] Paul C Blainey and Stephen R Quake. Digital mda for enumeration of total nucleic acid contamination. *Nucleic acids research*, 39(4):e19–e19, 2011.

[19] Henrike Niederholtmeyer, Viktoria Stepanova, and Sebastian J Maerkl. Implementation of cell-free biological networks at steady state. *Proceedings of the National Academy of Sciences*, 110(40):15985–15990, 2013.

[20] Cong Fang, Yanju Wang, Nam T Vu, Wei-Yu Lin, Yao-Te Hsieh, Liudmilla Rubbi, Michael E Phelps, Markus Müschen, Yong-Mi Kim, Arion F Chatziioannou, et al. Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples. *Cancer research*, 70(21):8299–8308, 2010.

[21] Joshua S Marcus, W French Anderson, and Stephen R Quake. Microfluidic single-cell mrna isolation and analysis. *Analytical chemistry*, 78(9):3084–3089, 2006.

[22] Tsun-Ming Tseng, Mengchu Li, Daniel Nestor Freitas, Travis McAuley, Bing Li, Tsung-Yi Ho, Ismail Emre Araci, and Ulf Schlichtmann. Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(8):1588–1601, 2017.

[23] Angela R Wu, Joseph B Hiatt, Rong Lu, Joanne L Attema, Neethan A Lobo, Irving L Weissman, Michael F Clarke, and Stephen R Quake. Automated microfluidic chromatin immunoprecipitation from 2,000 cells. *Lab on a Chip*, 9(10):1365–1370, 2009.

[24] Gurobi Optimization, Inc. Gurobi optimizer reference manual. http://www.gurobi.com.