Design-For-Testability of On-Chip Control in mVLSI Biochips

Potluri, Seetal; Pop, Paul; Madsen, Jan

Published in:
I E E E Design & Test

Link to article, DOI:
10.1109/MDAT.2018.2873448

Publication date:
2019

Document Version
Peer reviewed version

Citation (APA):
Potluri, S., Pop, P., & Madsen, J. (2019). Design-For-Testability of On-Chip Control in mVLSI Biochips. I E E E Design & Test, 36(1), 48-56. DOI: 10.1109/MDAT.2018.2873448

Design-For-Testability of On-Chip Control in mVLSI Biochips

Seetal Potluri, Paul Pop, Jan Madsen
Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Kongens Lyngby, Email:paupo@dtu.dk

***Abstract*—To enable mVLSI biochips for point-of-care applications, recent work has focused on reducing the number of off-chip pressure sources, using on-chip pneumatic control logic circuits fabricated using three-layer monolithic membrane valve technology. Since these on-chip pneumatic control logic circuits in turn control the fluidic operations, it is very important that they are fault-free, in order to avoid the failure of biochemical applications. For the first time, this paper proposes a design-for-testability (DFT) scheme to test for faults inside on-chip pneumatic control logic circuits, by adding observation pneumatic latches into the circuit.**

Keywords: mVLSI biochips, On-Chip Control, Manufacturing defects, Design-For-Testability ;

I. Introduction

The microfluidic Very Large Scale Integration (mVLSI) biochips allow miniaturization of biochemical processes, thus offering several advantages, similar to traditional VLSI [1], and thus being increasingly used in point-of-care applications. Physical defects can be introduced while manufacturing mVLSI biochips [2], which if used, results in failure of the biochemical application. This is a roadblock to their market penetration and researchers have started to propose automated testing of mVLSI biochips by converting their structure to a logic circuit model composed of Boolean gates, using air pressure as test signals, and using standard test pattern generation techniques, for post-fabrication testing [2].

The device size can be as small as $6\mu m$x$6\mu m$, and the mVLSI fabrication density as high as 1 million valves/cm^2 [3]. The fault count will be proportional to the device density, similar to VLSI [4]. This way, although the soft lithography technology used for fabricating mVLSI biochips has advanced similar to Moore's law [5], the off-chip pressure actuators and pumps are bulky, thereby limiting them to laboratory environments. To address this issue, recent work [6], [7] focused on reducing the number of off-chip pressure sources, using on-chip pneumatic control logic circuits fabricated using three-layer monolithic membrane valve technology. Since these on-chip pneumatic control logic circuits in turn control the fluidic operations, it is very important that they are fault-free, in order to avoid the failure of the biochemical application.

For the first time in literature, this paper proposes a design-for-testability (DFT) scheme to test for faults inside on-chip pneumatic control logic, to ensure high fault coverage, while keeping the cost overheads to a minimum. Our DFT scheme is novel compared to existing DFT schemes for VLSI [4] because mVLSI biochips have two parts (flow and control) and we need to test for faults in both the parts. The unique and novel feature of the proposed DFT scheme is that, apart from testing the on-chip control logic, the inserted pneumatic latches at the test points can be reused for testing the flow processor [2]. In this paper, we validate the proposed scheme on general-purpose on-chip control as an example. However, since the testing problem is orthogonal to the type of on-chip control, the proposed scheme is applicable to any kind of on-chip control.

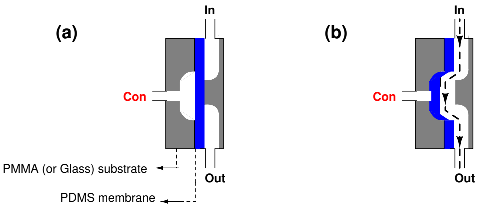


Figure 1. Microfluidic valve, where In, Out and Con refer to Input, Output and Control ports/signals respectively

II. Biochip architecture

Physically, the biochip can have multiple layers, but logically divided into two parts: *flow processor* and the *control circuit*. The basic building block of a biochip is a micro-mechanical valve, which restricts/permits the fluid flow, and hence used to manipulate the fluid in the flow processor. The microvalve is controlled using an external air pressure source, through a control pin.

Broadly speaking, there are two types of microvalves proposed in prior literature: normally open valves [8] and normally closed valves [9]. This paper considers normally closed valves, because they have been shown to exhibit good noise margins to implement pneumatic digital logic [7], [9]. However, the approach proposed in the paper can be extended to consider other valve technologies which implement pneumatic digital logic, e.g., normally-open valves [10]. A normally-closed valve is shown as a switch-like structure, in Figure 1(a), which has three layers.

The top and bottom layers are made of glass or polymethyl methacrylate (PMMA), and the middle layer is an elastic membrane made out of an elastomer such as polydimethylsiloxane (PDMS). The valve is normally closed, when vacuum is not applied to the control pin, named ”Con”. When vacuum (P_V) is applied to ”Con”, the PDMS membrane is pulled into a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2018.2873448, IEEE Design and Test

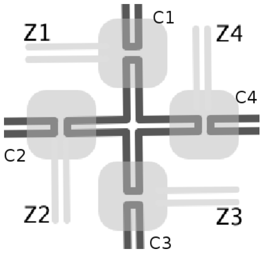


Figure 2. Omni-directional switch designed using microfluidic valves

Table I
ACTUATION TABLE FOR THE OMNIDIRECTIONAL SWITCH.

Phase	Z1	Z2	Z3	Z4
C1 - C2	1	1	0	0
C1 - C3	1	0	1	0
C1 - C4	1	0	0	1
C2 - C3	0	1	1	0
C2 - C4	0	1	0	1
C3 - C4	0	0	1	1

displacement chamber, allowing the fluid to flow between ”In” and ”Out”, as shown in Figure 1(b)).

A. Flow processor

Using valves and etched channels, more complex flow components can be built, such as switches, pumps, filters, heaters, storage units, detectors and separators [9]. Figure 2 shows the omni-directional switch, designed using 4 microfluidic valves $Z1$-$Z4$ and 4 microfluidic channels $C1$-$C4$. An actuation table shows the combination in which valves need to be actuated to make the fluid flow in a predetermined path. The actuation table of this omnidirectional switch, is shown in Table I. For example, if state of valves $Z1$ and $Z2$ are both 1, it indicates that corresponding valves are open and fluid flows freely between channels $C1$ and $C2$. This actuation table covers all the possible fluid flow scenarios of this omnidirectional switch.

B. Control circuit

Similarly, on-chip control logic circuits can be built using valves and etched channels. The on-chip control circuit can accept few primary inputs and can drive many more valves, essentially acting like an on-chip decompressor. Figure 3 shows a structure built using an etched channel and a normally closed valve. From figure 1, we know that when $P_I = P_V$, the valve opens, hence $P_O = P_A$. Also, when $P_I \neq P_V$, valve remains closed, hence $P_O = P_V$. Clearly, this structure represents a pneumatic NOT gate, as shown in the truth table in figure 3. It can also be noticed that the logical behavior of this valve resembles that of an nMOS transistor, where P_V is equivalent to supply voltage and P_A is equivalent to ground voltage. Hence, more complex pneumatic logic circuits can be built using normally closed valves (similar to nMOS-resistor logic). The reason to have on-chip control is to limit the number of external control pins. Control valve sharing [11] can also reduce the pin count, which is orthogonal to our problem. We do not perform valve sharing during synthesis of on-chip control circuits.

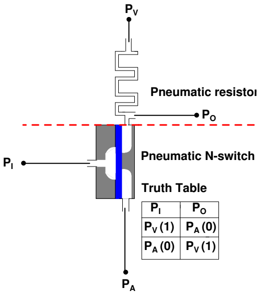


Figure 3. Pneumatic NOT gate built using pneumatic N-switch and resistor

Table II
TRUTH TABLE FOR THE LOGICAL CIRCUIT NEEDED TO CONTROL AN OMNIDIRECTIONAL SWITCH

Input			Output			
Q_3	Q_2	Q_1	Z1	Z2	Z3	Z4
0	0	0	1	1	0	0
0	0	1	1	0	1	0
0	1	0	1	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	1
1	1	0	X	X	X	X
1	1	1	X	X	X	X

III. ON-CHIP CONTROL SYNTHESIS

We start from a given biochip architecture, which is specified as a netlist for the flow processor. The control synthesis should ensure that the synthesized on-chip pneumatic control logic circuit is able to produce all the actuation signals necessary for the proper functioning of all the components in the flow processor for any biochemical application, i.e., the control is general purpose, given the particular architecture, not application-specific. In this context, general-purpose does not refer to the flow processor, rather it describes the capability of the on-chip control circuit. For example, an application-specific control will only use one route (path) on the flow processor for transport, and that too in only one direction. A general-purpose control will allow to control the transport such that all routes (paths) on the flow processor can be used in any direction.

This increases the control logic complexity, and hence the possibility of false positives in failure detection. However, the general-purpose property, means that we are not restricted to a single application (the biochip can run different applications) and results in versatile designs which can accommodate fault tolerance mechanisms and interactive testing based on reaction results [3].

With binary encoding, the number of control inputs needed reduces from E_a to $\lceil log_2(E_a) \rceil$, where E_a is the number of entries in the actuation table. After binary encoding of all the entries, the actuation table shown in Table I transforms to the control truth table shown in Table II, where Q_1, Q_2 and Q_3 are the binary-encoded (or Boolean) variables. Since Table I

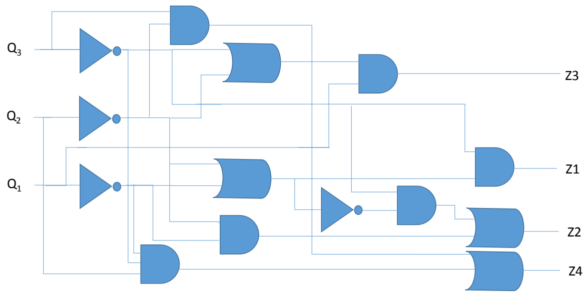


Figure 4. On-chip pneumatic control logic netlist synthesized for omni-directional switch shown in Figure 2 after logic minimization and gate-sharing

consists of only six entries, the last two entries among the eight entries in Table I, are shown as don't cares. Based on this control truth table, truth table, the following Boolean functions can be derived:

$$\begin{aligned}
Z1 &= \bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_3\bar{Q}_2Q_1 + \bar{Q}_3Q_2\bar{Q}_1 \\
Z2 &= \bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_3Q_2Q_1 + Q_3\bar{Q}_2\bar{Q}_1 \\
Z3 &= \bar{Q}_3\bar{Q}_2Q_1 + \bar{Q}_3Q_2Q_1 + Q_3\bar{Q}_2Q_1 \\
Z4 &= \bar{Q}_3Q_2\bar{Q}_1 + Q_3\bar{Q}_2\bar{Q}_1 + Q_3\bar{Q}_2Q_1
\end{aligned} \quad (1)$$

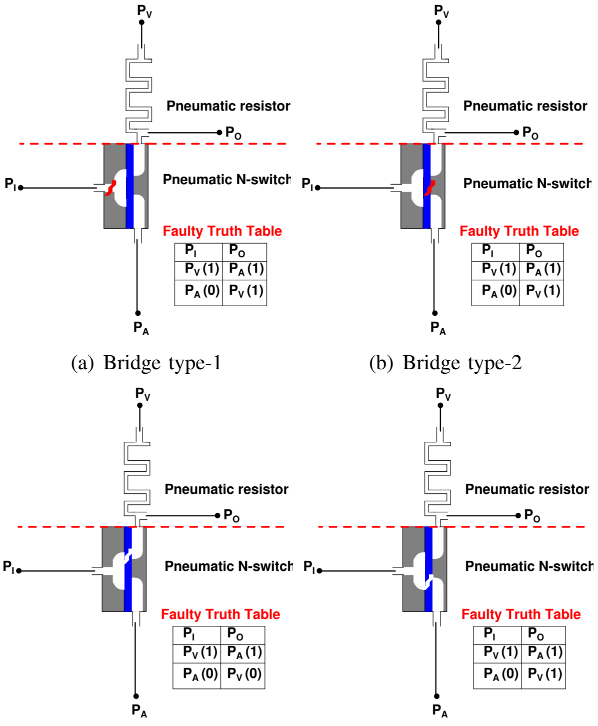
Prior to logic minimization, this circuit needed 24 2-input AND gates and 8 2-input OR gates. After X-filling, logic minimization and subsequent gate-sharing, the final circuit needed 4 NOT gates, 5 2-input AND gates, 1 3-input AND, and 4 2-input OR gates, as shown in Figure 4.

Finally, the logic minimized circuit is subject to library mapping to obtain a logic-level netlist in the target technology standard cell library (of monolithic membrane valve based logic gates). We use existing algorithms to solve these steps. Our proposed *Test Insertion Algorithm* is called after the *Library Mapping* step, and is presented in Sect. VII. The final two steps perform the physical synthesis of the control circuit. The details of the physical synthesis, for both the control and the flow parts, are available in [3].

IV. Defects and Fault Modelling

The fault models proposed in the past [2] address the defects for flow processor design with normally open valves. The types of defects that manifest in on-chip control circuits using normally closed valves are different, hence the fault models proposed for the flow processor [2] are inadequate.

Figure 5 shows the possible open and bridge defects in a normally closed valve and the corresponding faulty behavior of a NOT gate designed with such a defective valve. From figures 5(a), (b) and (d), we can notice that the corresponding faults manifest output stuck-1 fault behavior. However, figure 5(c) shows that the corresponding open defect does not manifest output stuck fault behavior. Thus, not all faults manifest gate stuck fault behavior. *Nevertheless, if we carefully analyze, we realize that if we test for both stuck-1 and stuck-0 faults at the output of NOT gate, automatically it will test the open defect shown in figure 5(c).*



(a) Bridge type-1
(b) Bridge type-2
(c) Open type-1
(d) Open type-2

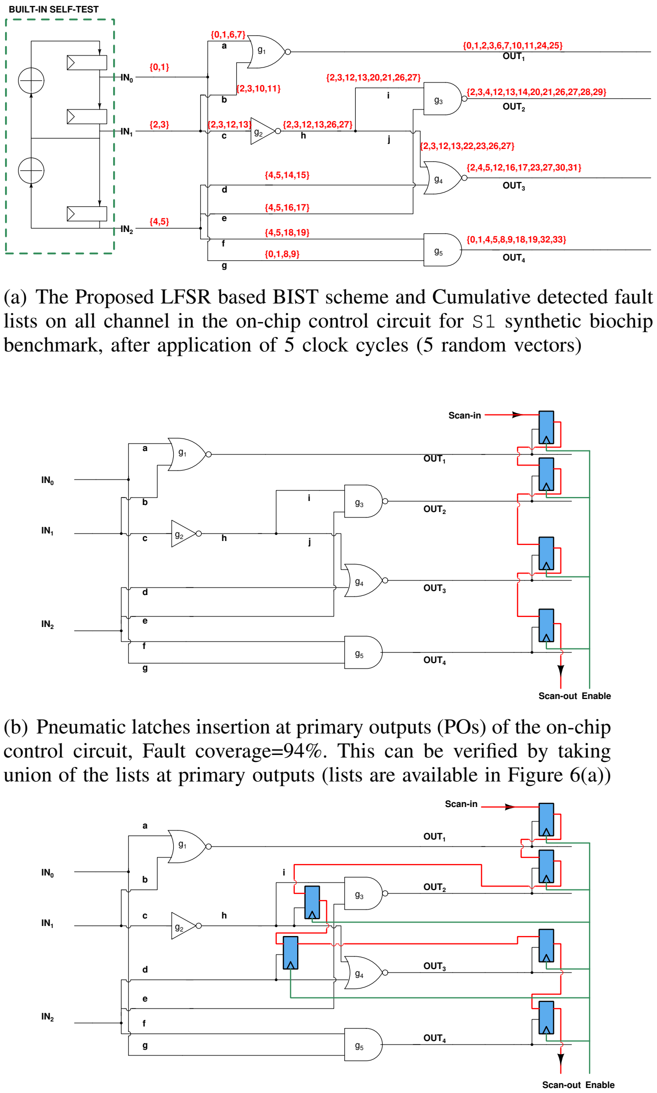
Figure 5. Types of bridge and open faults in pneumatic NOT gate and their faulty truth tables

The 2-input NOR and 2-input NAND gates also exhibit similar faulty behaviors. Due to lack of space, we skip the detailed faulty truth tables for all the possible internal defects for these complex gates, instead we summarize the faulty behaviors in Table III. Table III shows that if we test for both stuck-1 and stuck-0 faults at the output of the pneumatic gate, it is possible to detect all the possible bridges and opens for NOT, 2-input NOR and 2-input NAND gates. Thus, we

Table III
Fault behaviors and fault detection capability of stuck-at patterns for different pneumatic logic gates

Gate type	Defect type	Gate-level	Candidate stuck-at fault list for Defect detection
NOT	Bridge-1	O s-a-1	{O s-a-1}
	Bridge-2	O s-a-1	{O s-a-1}
	Open-1	I (buffer w.r.t input)	{O s-a-0, O s-a-1}
	Open-2	O s-a-1	{O s-a-1}
NOR	Bridge-1	-	{O s-a-0, O s-a-1}
	Bridge-2	-	{O s-a-0, O s-a-1}
	Open-1	-	{O s-a-0, O s-a-1}
	Open-2	O s-a-0	{O s-a-0}
NAND	Bridge-1	O s-a-1	{O s-a-1}
	Bridge-2	O s-a-1	{O s-a-1}
	Bridge-3	O s-a-1	{O s-a-1}
	Bridge-4	O s-a-1	{O s-a-1}
	Open-1	I_1 (buffer w.r.t input-1)	{O s-a-0, O s-a-1}
	Open-2	O s-a-1	{O s-a-1}
	Open-3	-	{O s-a-0, O s-a-1}
	Open-4	O s-a-1	{O s-a-1}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2018.2873448, IEEE Design and Test



(a) The Proposed LFSR based BIST scheme and Cumulative detected fault lists on all channel in the on-chip control circuit for `S1` synthetic biochip benchmark, after application of 5 clock cycles (5 random vectors)

(b) Pneumatic latches insertion at primary outputs (POs) of the on-chip control circuit, Fault coverage=94%. This can be verified by taking union of the lists at primary outputs (lists are available in Figure 6(a))

(c) Pneumatic latches insertion at selected test points in the on-chip control circuit, Fault coverage=100%. This can be verified by taking union of the lists at the test points (lists are available in Figure 6(a))

Figure 6. On-chip control for `S1` biochip

have verified that all open and bridge defects in an on-chip pneumatic control circuit composed of NOT, 2-input NAND and 2-input NOR gates can be detected with application of stuck fault patterns alone. Thus, by testing all the stuck faults in the gate-level on-chip control logic circuit, all the internal defects will automatically be tested.

V. Motivational Example

Figure 6(a) shows the pneumatic control logic circuit for `S1` biochip [3], and the set of faults detected on each channel (shown in red), using 5 randomly generated test vectors for this circuit using the Built-In-Self-Test (BIST) scheme. Since mVLSI biochips are not prevalently used today, maintaining the very expensive automatic test equipment (ATE) is not feasible, hence we propose to use BIST to generate and apply the pseudo-random test vectors. We propose to implement the BIST using a linear-feedback-shift-register (LFSR) [4], as shown in Figure 6(a), constructed using pneumatic flip-flops.

The sets (shown in red) in Figure 6(a) essentially quantify the set of faults detected at respective channels, after the application of all 5 random test vectors. In Figure 6(a), fault universe $U = \{0,1,2,3,4,5,6,7,8,9,10,11,12,13\ldots33\}$, thus $|U| = 34$. The outputs of on-chip pneumatic control logic circuit directly control the microvalves on the biochip. So, in order to test for faults inside the on-chip pneumatic control logic circuit, a straight-forward method is to insert pneumatic latches at the primary outputs. When pneumatic latches [7] are inserted only at primary outputs, as shown in Figure 6(b), $U^{'} = \{0,1\ldots11,12,13,14,16,17,18,19,20,21,23,24\ldots33\}$, where each candidate fault is assigned an integer, thus making $|U^{'}| = 31$, which translates to 94% fault coverage.

We go a step further to investigate if adding more pneumatic latches internal to the circuit, can improve the fault coverage. In this way, when pneumatic latches are inserted at primary outputs and some internal channels, as shown in Figure 6(c), $U^{'} = U$, indicating 100% fault coverage. This motivates the need to improve fault coverage, by inserting pneumatic latches at primary outputs and subset of channels inside the circuit. Now, the pneumatic latches added at primary outputs can be reused to apply pressure signals and test for faults inside the flow processor using the technique proposed in [2]. *This is the unique and novel feature of the proposed DFT scheme that allows us to test both the control and flow parts.*

VI. Problem Formulation

Let $C(LG,W)$ represent a pneumatic control logic circuit, where LG is the list of logic gates and W is the list of microfluidic channels inside C. Let the fault universe (list of faults to be covered) be U. Let $S_i \subset U$ be the list of faults detected on channel w_i, after the application of all the test vectors to the circuit, and let $S = \{S_1, S_2, \ldots S_N\}$, where $N = |W|$, the total wire count in the circuit. For any $W^{'} \subset W$, let $U(W^{'})$ be the set of faults detected on $W^{'}$.

Given C, set of test vectors and S, we are interested in choosing a subset of channels, $W^{'} \subset W$, for pneumatic latch insertion, such that the fault coverage $|U(W^{'})/|U| \times 100\%$ is maximized with minimum number of pneumatic latches. Since $U(W^{'}) = \cup_{i \in W^{'}} S_i$, which is a union of the elements from the subset of S, this translates to picking a minimum-sized collection of sets (subset) of S, such that the union of the collection is maximized, which is an instance of the *set cover problem*, which is NP-Hard. Since we are interested in union of fault sets to compute coverage, and since the problem under consideration is an instance of the NP-Hard set cover optimization problem, we use greedy set cover to solve the problem. We have not used ILP, since we are dealing with sets and not integers. Since minimum set cover and minimum hitting set are equivalent, we have decided to pursue only minimum set cover.

```
Algorithm 1: Proposed Algorithm for Test Point Insertion
  Input: Circuit $C(LG,W)$, Test Set $T$, Target fault
         coverage $\eta$
  Output: Set of Test Points returned by GreedySetCover
1 Let $S_i$ be the set of faults detected on wire $w_i \in W$ over
  the entire test set $T$;
2 Perform deductive fault simulation [4] of $C$ for all test
  vectors and compute $S_i$ for each $w_i \in W$;
3 Let $U$ be the set of all possible faults inside $C$;
4 Let $S = \{S_1, S_2, \ldots S_N\}$, where $N = |W|$;
5 Solve GreedySetCover($U$,$S$,$\eta$);
```

```
Algorithm 2: GreedySetCover($U$,$S$,$\eta$)
  Input: Fault universe $U$, Fault simulation collection of
         sets $S$, Target fault coverage $\eta$
  Output: $TP$
1 Let $TP = \emptyset$;
2 while $|TP| < \eta * |U|$ do
3     Let $max\_set \in S$ be an unvisited set that contains the
      largest number of uncovered faults from $U$;
4     $TP = TP \cup max\_set$;
5     Mark all uncovered faults in $U$, that are present in
      $max\_set$ as covered;
```

VII. The Proposed Test Point Insertion Algorithm

Algorithm 1 shows the proposed test point insertion, that uses a greedy set cover approach to maximize fault coverage and to minimize the number of test points. The input Test Set contains the test vectors generated by the automatic test pattern generation algorithm [4]. Algorithm 1 uses Algorithm 2 internally. Algorithm 2 takes the fault universe U and fault coverage collection of sets S as inputs and outputs a set of test points TP inside the circuit, which achieves the desired fault coverage η. This algorithm begins with a null set TP in line 1, when all sets in S as unvisited. Lines 4-6 *greedily* pick the largest sized unvisited set in S, and marks it as visited, and appends all of its elements to TP (union operation). When the number of faults in TP achieves the desired fault coverage η, the while loop shown in step 2 terminates and after this point TP is returned by *GreedySetCover* algorithm.

VIII. Experimental Results

For all our experiments, we have used an Intel Core i7 8-core CPU, running at 2.67 GHz with 5.8 GB of memory. The implementation has two phases:

- list generation through deductive fault simulation; and
- solving the greedy set cover algorithm.

The list generation phase was implemented as a *C* program and the greedy set cover is implemented using the *python*3.4 language. The implementation was tested on five benchmarks obtained from [3]: `S1`, `PCR1`, `PCR2`, `PCR3` and `EA`. Information about the benchmark complexity in terms of the number of flow components, flow connections, control pins, and control connections is available in [3]. Optimal test point insertion in the on-chip control circuit for `S1` benchmark is already discussed in Sect. V, and shown in Figure 6(c) in the same section. The experimental results obtained for the `PCR1`, `PCR2`, `PCR3` and `EA` are shown in Tables V, VI, VII and VIII respectively.

In these four tables, a "—" entry indicates that the corresponding fault coverage is not achievable. The following important observations can be made from these tables:

- the "—" entries indicate that even if we are allowed to insert as many test points as possible, it is not possible to achieve a fault coverage >60% and >90% with *1 random vector* and *2 random vectors* respectively;
- the absence of "—" entries in >60% rows for more than 1 random vector and in >90% rows for more than 2 random vectors, indicates that some of the faults get detected only when more random vectors are applied; and
- Given a target fault coverage, there is a trade-off between the number of random vectors applied and the number of test points necessary to achieve the coverage. This is evident from the tables, as the number of test points for a given target coverage, monotonically decrease with increasing the number of random vectors;

Additionally, for a given number of random vectors, fault coverage improves by increasing the number of test points. In Figure 6(c) of Sect. V, we have noted that the fault coverage at test points is 6% higher than the fault coverage at primary outputs alone. Table IV performs this analysis for all the remaining benchmarks. From this table, it can be noted that across all benchmarks, there is consistent improvement in maximum fault coverage (MFC) at test points as compared to MFC at primary outputs of the on-chip control circuit. Additionally, from this table, it can be noted that close to 95% fault coverage is achieved for all benchmarks. About 15-20% of the device area is required for adding the pneumatic latches and routing them. For all the benchmarks, the free space available was found to be actually sufficient for this purpose [3]. *Thus, the proposed DFT scheme improves fault coverage with no area overheads.*

A. Eliminating Area Overhead through Imaging

We have found for benchmark circuits, that pneumatic flip-flop insertion does not add any area-overhead, due to the available free space around the flow processor. While this was true for the specific benchmarks considered, this cannot be guaranteed in theory. An alternative to overcome this limitation, is to use imaging to detect the faulty behavior of the logic gates. Since these logic gates are fabricated with transparent material and their response times are typically in seconds [7], it is possible to imaging techniques to detect them. The imaging and video processing techniques were shown to be very successful for online detection and correction of faults in digital microfluidic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2018.2873448, IEEE Design and Test

Table IV
MAXIMUM FAULT COVERAGE (MFC) AFTER SOLVING *greedy set cover*

Number of Random vectors	MFC at primary outputs				MFC at test points			
	PCR1	PCR2	PCR3	EA	PCR1	PCR2	PCR3	EA
1	29.69%	27.19%	24.20%	19.54%	48.44%	46.93%	48.02%	47.80%
2	46.87%	45.18%	39.29%	30.77%	70.31%	71.93%	71.43%	68.08%
5	79.69%	75.0%	63.10%	51.69%	95.31%	91.67%	83.73%	86.0%
10	92.19%	82.46%	85.71%	68.46%	96.88%	92.98%	94.84%	91.23%
20	92.19%	82.46%	93.25%	84.69%	96.88%	92.98%	95.64%	94.77%

Table V
NUMBER OF TEST POINTS FOR *PCR*1 BENCHMARK (64 CANDIDATE FAULTS), FOR A GIVEN FAULT COVERAGE

Fault Coverage	1 Random vector	2 Random vectors	5 Random vectors	10 Random vectors
>30%	6	2	2	1
>60%	—	6	3	3
>90%	—	—	8	5

Table VI
NUMBER OF TEST POINTS FOR *PCR*2 BENCHMARK (228 CANDIDATE FAULTS), FOR A GIVEN FAULT COVERAGE

Fault Coverage	1 Random vector	2 Random vectors	5 Random vectors	10 Random vectors
>30%	12	3	2	1
>60%	—	31	10	3
>90%	—	—	31	22

Table VII
NUMBER OF TEST POINTS FOR *PCR*3 BENCHMARK (252 CANDIDATE FAULTS), FOR A GIVEN FAULT COVERAGE

Fault Coverage	1 Random vector	2 Random vectors	5 Random vectors	10 Random vectors
>30%	19	6	3	1
>60%	—	40	12	5
>90%	—	—	—	18

Table VIII
NUMBER OF TEST POINTS FOR *EA* BENCHMARK (1300 CANDIDATE FAULTS), FOR A GIVEN FAULT COVERAGE

Fault Coverage	1 Random vector	2 Random vectors	10 Random vectors	20 Random vectors
>30%	48	15	3	2
>60%	—	143	14	6
>90%	—	—	147	38

biochips [12]. We propose to use the same in the context of testing faults within on-chip control circuits for mVLSI biochips. The quality/cost of camera required depends on the desired fault coverage level. This can be better understood as follows:

- We have seen earlier that increasing test points, increases fault coverage. Similarly, the more the number of test points observed using imaging, the more will be achieved level of fault coverage.
- We have seen earlier that increasing test points increases area-overhead. Similarly, as more test points are introduced, a finer camera resolution will be required for the imaging, thus increasing the price.

IX. CONCLUSIONS

For the first time in literature, we have developed a DFT scheme for testing of on-chip pneumatic control circuits for mVLSI biochips. We have shown that the problem of attaining high fault coverage through test point insertion is an instance of the *set cover problem*, which is NP-hard and used a greedy heuristic to solve the problem. The unique and novel feature of the proposed DFT scheme is that it allows us to test both the control and flow parts of a biochip, with no area overheads. The efficacy of the proposed algorithms were demonstrated on biochip benchmarks.

REFERENCES

[1] D. Mark et al, “Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications,” *CSR*, vol. 39, pp. 1153–1182, 2010.

[2] K. Hu et al, “Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration,” in *IEEE TCAD, Vol. 33, No. 10, October 2014*, pp. 1463–1475, 2014.

[3] P. Pop et al, *Microfluidic Very Large Scale Integration (VLSI): Modeling, Simulation, Testing, Compilation and Physical Synthesis*. Springer International Publishing, 2016.

[4] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital systems testing and testable design.* Computer Science Press, 1990.

[5] I. E. Araci et al, “mVLSI with integrated micromechanical valves,” in *Lab on Chip*, vol. 12, pp. 1463–1475, 2012.

[6] E. C. Jensen et al, “Micropneumatic digital logic structures for integrated microdevice computation and control,” *IEEE Journal of Micromechanical Systems*, vol. 16, no. 6, pp. 1378–1385, 2007.

[7] P. N. Duncan, T. V. Nguyen, and E. E. Hui, “Pneumatic oscillator circuits for timing and control of integrated microfluidics,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18104–18109, 2013.

[8] T. Thorsen et al, “Microfluidic large-scale integration,” *Science*, vol. 298, no. 5593, pp. 580–584, 2002.

[9] W. H. Grover et al, *”Monolithic Membrane Valves and Pumps” chapter in Lab-on-Chip Technology*. Caister Academic Press, 2009.

[10] J. A. Weaver et al, “Static control logic for microfluidic devices using pressure-gain valves,” *Nature Physics*, pp. 218–223, 2010.

[11] A. Schneider et al, “Pin-count reduction for continuous flow microfluidic biochips,” *Microsystem Technologies*, pp. 483–494, 2018.

[12] K. Hu, B.-N. Hsu, A. Madison, K. Chakrabarty, and R. Fair, “Fault detection, real-time error recovery, and experimental demonstration for digital microfluidic biochips,” in *proceedings of the conference on design, Automation and Test in Europe*, pp. 559–564, EDA Consortium, 2013.