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

Potluri, Seetal; Pop, Paul; Madsen, Jan

Published in:
IEEE Design & Test

Link to article, DOI:
10.1109/MDAT.2018.2873448

Publication date:
2019

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
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 $\delta_{\mu m} \times \delta_{\mu m}$, 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.

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...
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 \( P_1 = P_4 \), the valve opens, hence \( P_0 = P_A \). Also, when \( P_1 \neq P_4 \), valve remains closed, hence \( P_0 = P_A \). Clearly, this structure represents a pneumatic NOT gate, as shown in the truth table in figure 3.

We do not perform valve sharing during synthesis of on-chip control circuits.

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 \( \log_2(E_a) \), 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
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:

\[
\begin{align*}
Z_1 &= \bar{Q}_3 \bar{Q}_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_2 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_3 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_4 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 Q_1
\end{align*}
\]

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 consisted 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, the following Boolean functions can be derived:

\[
\begin{align*}
Z_1 &= \bar{Q}_3 \bar{Q}_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_2 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_3 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 \bar{Q}_1 \\
Z_4 &= \bar{Q}_3 Q_2 Q_1 + \bar{Q}_3 Q_2 \bar{Q}_1 + Q_3 Q_2 Q_1
\end{align*}
\]

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 faulty truth tables of 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 NOT gate, automatically it will test the open defect shown in figure 5(c).

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 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
feasible, hence we propose to use BIST to generate and apply test vectors. Since biochip [3], and the set of faults detected on each channel (shown in red), using 5 randomly generated test vectors for biochip [3], and the set of faults detected on each channel, after the application of all 5 random test vectors. In Figure 6(a), fault universe $U = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, \ldots, 33\}$, 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), $U' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 23, 24, \ldots, 33\}$, 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 $U(W') = \bigcup_{i \in W'} S_i$, 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 $|TP| < \eta \times |U|$ do
3. Let $max\text{-}set \in S$ be an unvisited set that contains the largest number of uncovered faults from $U$;
4. $TP = TP \cup max\text{-}set$;
5. Mark all uncovered faults in $U$, that are present in $max\text{-}set$ 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 $\eta$. 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 $\eta$, 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 python3.4 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
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.

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.

**TABLE IV**

<table>
<thead>
<tr>
<th>Number of Random vectors</th>
<th>MFC at primary outputs</th>
<th>MFC at test points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCR1</td>
<td>PCR2</td>
</tr>
<tr>
<td>1</td>
<td>29.69%</td>
<td>27.19%</td>
</tr>
<tr>
<td>2</td>
<td>46.87%</td>
<td>45.18%</td>
</tr>
<tr>
<td>5</td>
<td>79.69%</td>
<td>75.00%</td>
</tr>
<tr>
<td>10</td>
<td>92.19%</td>
<td>82.46%</td>
</tr>
<tr>
<td>20</td>
<td>92.19%</td>
<td>82.46%</td>
</tr>
</tbody>
</table>

**TABLE V**

<table>
<thead>
<tr>
<th>Fault Coverage</th>
<th>Number of Test Points for PCR1 Benchmark (64 Candidate Faults), for a Given Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Random vector</td>
</tr>
<tr>
<td>&gt;30%</td>
<td>6</td>
</tr>
<tr>
<td>&gt;60%</td>
<td>—</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>—</td>
</tr>
</tbody>
</table>

**TABLE VI**

<table>
<thead>
<tr>
<th>Fault Coverage</th>
<th>Number of Test Points for PCR2 Benchmark (228 Candidate Faults), for a Given Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Random vector</td>
</tr>
<tr>
<td>&gt;30%</td>
<td>12</td>
</tr>
<tr>
<td>&gt;60%</td>
<td>—</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>—</td>
</tr>
</tbody>
</table>

**TABLE VII**

<table>
<thead>
<tr>
<th>Fault Coverage</th>
<th>Number of Test Points for PCR3 Benchmark (252 Candidate Faults), for a Given Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Random vector</td>
</tr>
<tr>
<td>&gt;30%</td>
<td>19</td>
</tr>
<tr>
<td>&gt;60%</td>
<td>—</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>—</td>
</tr>
</tbody>
</table>

**TABLE VIII**

<table>
<thead>
<tr>
<th>Fault Coverage</th>
<th>Number of Test Points for EA Benchmark (1300 Candidate Faults), for a Given Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Random vector</td>
</tr>
<tr>
<td>&gt;30%</td>
<td>48</td>
</tr>
<tr>
<td>&gt;60%</td>
<td>—</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>—</td>
</tr>
</tbody>
</table>

**REFERENCES**