Design-for-Testability for Continuous-Flow Microfluidic Biochips

Chunfeng Liu[1,4], Bing Li[1], Tsung-Yi Ho[2,4], Krishnendu Chakrabarty[3,4], Ulf Schlichtmann[1]

[1]Chair of Electronic Design Automation, Technical University of Munich, Germany [2]National Tsing Hua University, Hsinchu, Taiwan
[3]Department of ECE, Duke University, Durham, NC, USA [4]Institute for Advanced Study, Technical University of Munich, Germany
{chunfeng.liu, b.li, ulf.schlichtmann}@tum.de, tyho@cs.nthu.edu.tw, krish@ee.duke.edu

ABSTRACT

Flow-based microfluidic biochips are gaining traction in the microfluidics community since they enable efficient and low-cost biochemical experiments. These highly integrated lab-on-a-chip systems, however, suffer from manufacturing defects, which cause some chips to malfunction. To test biochips after manufacturing, air pressure is applied to input ports of a chip and predetermined test vectors are used to change the states of microvalves in the chip. Pressure meters are connected to the output ports to measure pressure values, which are compared with expected values to detect errors. To reduce the cost of the test platform, the number of pressure sources and meters should be reduced. We propose a design-for-testability (DFT) technique that enables a test procedure with only a single pressure source and a single pressure meter. Furthermore, the valves inserted for DFT share control channels with valves in the original chip so that no additional control signals are required. Simulation results demonstrate that this technique can generate efficient chip architectures for single-source single-meter test in all experiment cases successfully to reduce test cost, while the performance of these chips in executing applications is still maintained.

1 INTRODUCTION

Microfluidic biochips are revolutionizing the traditional experiment flow in biochemical and pharmaceutical laboratories with their high execution efficiency and miniaturized fluid manipulation [1, 2]. In such a chip, microdevices are built to execute specific operations, such as mixing and detection. Between devices, microchannels are integrated to transport fluid samples following the protocol of a biochemical application. All these functions are performed at the nanoliter level and controlled by a microcontroller without human intervention. With their efficiency and reliability, these miniaturized and automated chips have been reshaping many fields such as pharmacy, biotechnology and health care. For example, genomic bioassay protocols, such as nucleic-acid isolation, DNA purification and DNA sequencing, have successfully been demonstrated with microfluidic biochips in recent years. In addition, this technology has also attracted a lot of commercial attention, e.g., from Illumina [3] and Agilent [4].

Structurally, a continuous flow-based biochip is built from basic components such as microchannels and microvalves, henceforth called channels and valves for simplicity. Flow channels are tiny transportation paths between devices through which reaction samples and reagents are moved between different devices. Above flow channels, control channels are built to conduct air pressure to intersections of flow channels and control channels to form valves, as illustrated in Figure 1(a). These channels are built from elastic materials, so that air pressure in a control channel squeezes the flow channel downwards to block the movement of the fluid sample underneath. If the pressure in the control channel is removed, the fluid sample can resume its movement.

With valves as basic controlling components, complex devices and a transportation network can be constructed. Figure 1(b) shows a mixer (reaction loop) surrounded by flow channels (green), control channels (yellow and red) and valves (yellow and red blocks). These channels and valves together form a network similar to the road transportation system. If fluid channels should cross, valves are built to form a switch. At any moment, only two of the valves of a switch open to direct fluid transportation; the other valves are closed to avoid fluid contamination.

Biochips are used to execute biochemical applications, which are usually specified using sequencing graphs $\mathcal{G} = (\mathcal{O}, \mathcal{E})$, such as in Figure 2, where $\mathcal{O}$ is the set of nodes and $\mathcal{E}$ is the set of edges. A node $O_i \in \mathcal{O}$ in the sequencing graph represents an operation, whose type and duration are specified by the user. An edge $e_{ij} \in \mathcal{E}$ from O_i to O_j in the sequencing graph specifies that O_i must be executed before O_j and the result of O_i is the input of O_j.

In a biochip, the number of dedicated devices is usually smaller than the number of operations in an application, so that the operations need to be executed with time multiplexing. Accordingly, intermediate reaction results need to be moved between different devices, leading to different fluid samples being transported inside the channel network between devices. For example, if O_i and O_j in a sequencing graph are executed by different devices, the result of O_i must be transported to the device executing O_j through channels. If the target device executing O_j is still occupied, the result of O_i should be stored temporarily, e.g., inside the channel, leading to a distributed storage [5, 6]. The overall execution time of an application is determined by the time when the last operation is finished. This execution time is one of the major performances in biochip design.

As in the semiconductor industry, design automation tools are also needed to support the development of microfluidic biochips. Since the synthesis flow of biochips is similar to the synthesis flow for integrated circuits, researchers in the electronic design automation community have started to expand into this area in recent years [7, 8]. Among them, the method in [9] proposes a top-down flow to generate a biochip architecture while minimizing the execution time of a bioassay. The flow channel routing problem considering obstacles is solved with an algorithm based on a rectilinear Steiner minimum tree in [10]. To avoid contamination, path searching is used in [11] to wash devices and channel segments. In addition, the method in [12] minimizes pressure-propagation delay in the control layer to reduce the response time of valves and synchronize their actuation. Furthermore, flow layer and control layer codesign is investigated in [13] to achieve a valid routing iteratively, and length-matching in routing control channels is considered in [14] as well.

Similar to ICs, manufactured biochips may also contain defects. For example, a channel might be blocked so that fluid samples cannot be moved through it. In addition, a valve may not be closed properly, leading to fluid contamination when executing an application. Accordingly, test of biochips has become another important research topic recently. In [15], several types of defects after manufacturing of flow-based biochips are analyzed and the test problem is converted into an ATPG equivalence in the IC design domain to generate test vectors. In addition,

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196025

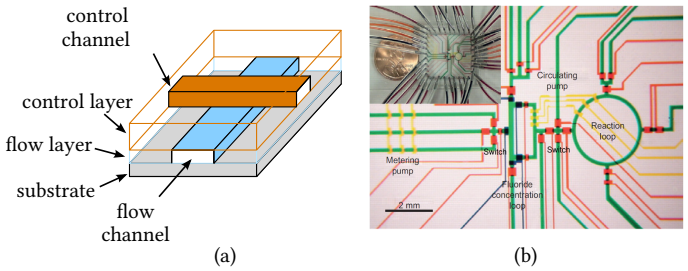

Figure 1: Components and structure of flow-based biochips. (a) Valves constructed at intersections of flow/control channels. (b) Biochip containing a mixer surrounded by a transportation channel network [18].

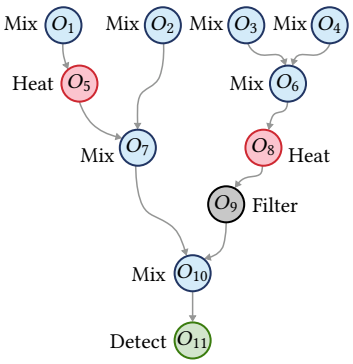

Figure 2: Sequencing graph of a biochemical application [9].

the method in [16] examines the relationship between test vectors and pressure measurements and constructs test vectors accordingly. Furthermore, [15] is expanded in [17] to improve the testability of a chip by modifying it after untestable scenarios are observed. These methods, however, do not consider cost reduction of the overall test platform, where multiple test devices are still needed. In addition, the modification of the chip architecture for testability may affect the execution efficiency of an application on the chip; despite its importance, this problem has not been considered in existing test solutions.

In this paper, we propose a design-for-testability (DFT) method to reduce the cost of the overall test platform. In addition, we introduce a codesign framework that improves testability and execution efficiency of biochip architectures simultaneously. The major contributions of this paper are as follows.

- We propose a design-for-testability method for flow-based biochips that drastically reduces the number of required test devices, e.g., pressure sources and meters. By adding additional channels and valves at proper locations, test vectors only rely on a single pressure source and a single pressure meter, so that both the cost and the complexity of the test platform are reduced.
- The newly introduced channels and valves are also used to improve the execution efficiency of applications. When adding channels and valves to improve testability, the configuration that provides the shortest execution time of the application is selected.
- The control channels corresponding to the new valves can also be shared with those for the valves originally existing in the biochip, so that the control logic does not need any change. Consequently, no additional control port is required. This sharing of control ports still allows a valid set of test vectors, while the execution efficiency of the application is maintained.
- The new chip architecture is constructed automatically using a two-level particle swarm optimization (PSO) method to improve a given chip architecture and generate efficient schedules for applications. Test vectors and the valve sharing scheme are returned as the output of this PSO method.

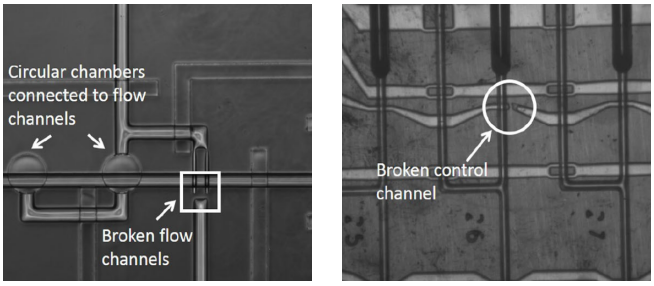

Figure 3: Manufacturing defects in continuous-flow biochips [15]. (a) Broken flow channel. (b) Broken control channel.

The rest of this paper is organized as follows. In Section 2, we explain the motivation and formulate the design-for-testability problem. The modification of a given chip architecture for DFT is explained in Section 3 and the proposed two-level PSO framework is described in detail in Section 4. Simulation results are reported in Section 5. Conclusions are drawn in Section 6.

2 MOTIVATION AND PROBLEM FORMULATION

In flow-based microfluidic biochips, flow channels and control channels are etched upon a substrate from PDMS through a process that includes molding and assembling [19]. In this multiple-step process, defects may appear in the manufactured chips. For example, flow channels may be broken during the molding process; flow and control layers may also be aligned improperly so that the pressure conducted through control channels cannot reach the corresponding valves, leading to valves that cannot be closed. Figure 3 shows cases with broken flow and control channels [15], two typical defects after manufacturing, leading to defects such as flow channels that are blocked and valves that cannot be closed.

To identify biochips with defects after manufacturing, a given set of test vectors needs to be applied, each of which is a combination of valve states. In the test procedure, air pressure sources are connected to some ports of the chip. A test vector is then applied to open and close the valves to create flow paths inside the chip to test the defects where channels are blocked or valves cannot be opened. Pressure meters are attached to other ports of the chip and the measured pressure results are compared with a predetermined set of measurements. If discrepancy exists, the chip is then reported to have defects. Reciprocally, to test the defects that valves cannot be closed, some valves are closed to separate the pressure sources and meters. If pressure can still be measured, a defect is thus detected. In this test procedure, usually air pressure instead of fluid is used to keep the chip from being contaminated before they are sold on the market.

The test procedure can be explained using Figure 4(a), where the biochip has three ports and six valves. Since a valve allows pressure propagation in either direction, the ports of a biochip are equivalent in the test procedure, so that each of them can be connected to either a pressure source or a pressure meter. In Figure 4(a), one of the ports of the biochip is connected to a pressure source and the other two are connected to pressure meters.

To detect the defects in which valves cannot be opened, two test vectors, path P1 and path P2, are generated for the biochip in Figure 4(a). When such a test vector is applied, all the other valves that are not on the path are closed. If a valve on the path cannot be opened due to manufacturing defects, the pressure meters cannot measure any pressure, because no other path from the pressure source to the meter exists. Similar to test paths, test vectors also contain cuts to verify whether valves can be opened correctly. For example, if the valves in

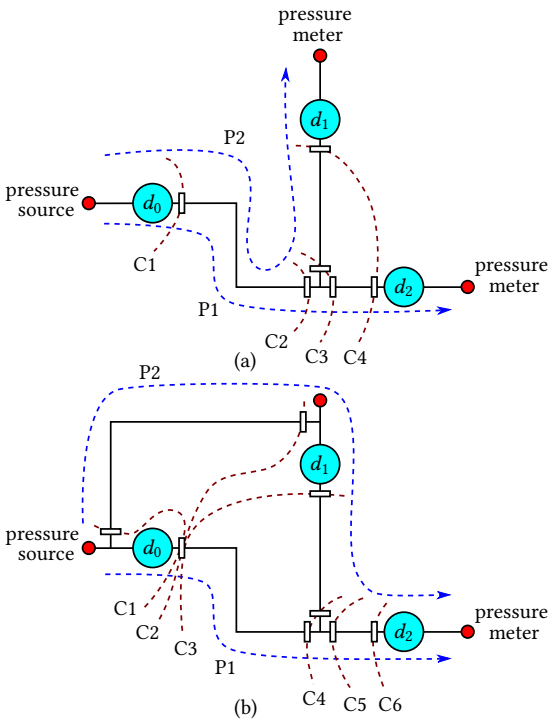

Figure 4: Test vectors and design-for-testability. (a) Test paths and cuts to detect stuck-at-0 and stuck-at-1 defects. (b) Single-source single-meter test.

one of the four cuts C1–C4 in Figure 4(a) are closed, the chip is separated into two parts and the pressure from the source should not reach any pressure meter. If a pressure meter still reads a valid pressure value, a defect is then detected. For convenience, we call a valve that cannot be opened or a channel that is blocked a stuck-at-0 defect, and a valve that cannot be closed a stuck-at-1 defect. Besides these defects, there are also leakage defects between flow channels and control channels, which also affect the function of a chip and can be tested similarly [15]. In this paper, we will use only stuck-at-0 and stuck-at-1 defects to demonstrate the proposed design-for-testability method.

In Figure 4(a), one pressure source and two pressure meters are required. These mechanical devices are cumbersome and the installation as well as their connection to the biochip under test makes the test procedure error-prone. Therefore, it is beneficial to reduce the number of such devices to the minimum in the test platform. To achieve this reduction, the architecture of the biochip in Figure 4(a) can be modified to add more channels and valves, so that only one pressure source and one meter are required, as shown in Figure 4(b). With this reduction, test paths P1-P2 and cuts C1-C6 can still be generated successfully to detect manufacturing defects. Obviously this modification introduces more channels and valves into the chip. However, the cost of the chip does not increase noticeably, because biochips are manufactured using masks as for IC manufacturing, and a large number of valves can already be built in a chip cost-effectively [1]. The increased number of test vectors is also affordable, since test time is still not the first concern in state-of-the-art biochip design [15].

The valves introduced in the modified chip architecture still need to be controlled by control ports connected to them, as illustrated in Figure 1. In the case where additional independent control ports can be provided, the two valves and channels in Figure 4(b) provide new resources for the execution of applications. Therefore, the performance of the biochip can be improved accordingly. If, on the other hand, the control logic is not capable of providing additional independent control ports, the new valves introduced to increase testability should share a control port together with another existing valve in the original biochip. This sharing still needs to allow a valid set of test vectors to be generated to test all the stuck-at-0 and stuck-at-1 defects.

When an application is executed by the modified biochip, it should provide similar performance compared with the version before modification. The sharing of control logic between a newly added valve and an existing valve, however, limits the flexibility in transporting fluid results between devices, because contamination may occur due to the valves that are forced to open together with another valve. Similarly, the forced close state of shared valve increases the chance of blocking in fluid transportation, thus also leading to a low execution efficiency of the application.

With the discussion above, we can then formulate the design-for-testability problem as follows.

Given: A biochip architecture with the application to be executed.

Output: An augmented biochip architecture for testability; a set of test vectors considering valve sharing; a schedule of the application on the augmented biochip architecture.

Objectives: The augmented architecture requires only a single pressure source and a single pressure meter for testing manufacturing defects; the execution efficiency of the application should be maintained at the same level.

3 CONSTRUCTING DESIGN-FOR-TESTABILITY BIOCHIP ARCHITECTURE

In this section, we explain the strategy and the implementation to generate an augmented biochip architecture and the corresponding test vectors.

As observed in Figure 4, a biochip with multiple ports can obtain single-source single-meter testability by adding additional DFT channels and valves. In the proposed method, the locations of these channels and valves are determined by mapping the input chip architecture to a virtual connection grid, as shown in Figure 5. In this mapping, devices are assigned to nodes and channels to edges in the grid, while keeping the original topology of the chip unchanged. After this mapping, the nodes and edges that have not been occupied in the connection grid represent the possibilities where additional channels and valves can be built. In the mapping in Figure 5, valves need not to be mapped to the nodes explicitly, because they are needed only at the inputs/outputs of devices and the crossing points between channels and they are tested together with valves.

The testability of stuck-at-0 defects of a biochip requires that for each valve or channel, there is a path from the single pressure source to the single pressure meter. In the proposed method, we use this requirement to identify the locations of new channels and valves. Afterwards, test cuts are generated from the augmented architecture to test stuck-at-1 defects.

For a given biochip architecture, there might be multiple possibilities to add channels and valves to implement the single-source single meter testability. In the following, a feasible solution of this testability problem is called a *DFT configuration*. In the proposed method, we identify the DFT configurations using Integer Linear Programming (ILP) programming.

Assume there is a path p_r between nodes n_s and n_t, which represent the locations of the external ports on the connection grid. We use a 0-1 variable $e_{j,r}$ to represent whether the edge e_j in the connection grid is on the path p_r, and the 0-1 variable $n_{i,r}$ to represent whether the node n_i is on the path p_r. For nodes $n_{s,r}$ and $n_{t,r}$, only one of the four edges incident to it can be covered by the path p_r. At each other node n_i on the path, exactly two edges incident to it are covered. Accordingly, we

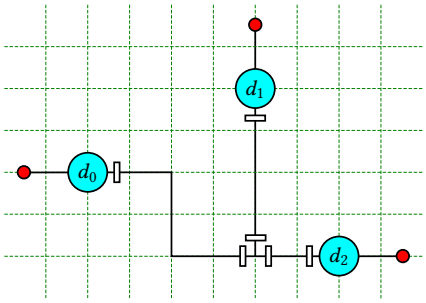

Figure 5: Mapping a biochip onto a virtual connection grid.

can construct the path with the following constraints,

$$\sum_{e_j \in E_i} e_{j,r} = 2n_{i,r}, \quad \forall n_i \in N, n_i \neq n_s, n_t \tag{1}$$

$$\sum_{e_j \in E_i} e_{j,r} = 1, \quad \forall n_i = n_s, n_t \tag{2}$$

where E_i is the set of edges incident to the node n_i and N is the set of all the nodes in the connection grid.

Since the channel segments occupied by the original chip must appear in the new architecture, the constraints for these edges can be written as

$$\sum_{p_r \in P} e_{j,r} \geq 1, \quad \forall e_j \in E_o \tag{3}$$

where P is the set of all test paths; E_o is the set of edges occupied by the original channels in the connection grid.

When generating the DFT architecture of the chip, we minimize the number of added channels. We use a 0-1 variable s_j to represent whether the edge e_j in the connection grid should be kept in the final chip. If any test path covers e_j, s_j must be 1, so that the relation between s_j and e_j can be established as

$$s_j \geq e_{j,r}, \quad \forall p_r \in P, \quad \forall e_j \in E \backslash E_o \tag{4}$$

where P is the set of all test paths and $E \backslash E_o$ represents the set of edges in the connection grid that are not covered by the original chip.

Finally, the architecture of the DFT chip can be determined by solving the following optimization problem

$$\text{minimize} \quad \sum_{e_j \in E \backslash E_o} s_j \tag{5}$$

$$\text{subject to} \quad (1)\text{–}(4). \tag{6}$$

In this optimization, the number of potential test paths $|P|$ should be determined in advance. This number should be large enough to cover all the channels in the augmented architecture. In the implementation, we started by setting this number to 2, and increased it by 1 in the next iteration when the current number produces no valid result. Another issue is that the formulation above assumes that starting and ending nodes of paths are given. Although any two ports in the chip are feasible test ports in the final DFT architecture, we used the two ports between which the distance is the largest to generate long instead of short test paths, so that more channels and devices in the original chip can be covered by an individual path potentially. The last issue of the formulation above is that loops may be generated on the paths. These loops need to be excluded to avoid false coverage of the original channels and devices. In the proposed method, we used the technique in [16] to exclude these loops.

In generating DFT valves and channels above, the test paths to detect stuck-at-0 defects have been created simultaneously. The stuck-at-1 defects are detected by test cuts applied to the chip. A cut is composed of a set of valves which separate the port connected to the pressure source and the port connected to the pressure meter during test. After test paths are created for single-source single-meter test, test cuts can always be created successfully because in the worst case test paths can be blocked individually to form cuts. The problem to find the minimum set of test cuts for a given biochip is a complementary problem of the test path generation so that the details are omitted due the page limit.

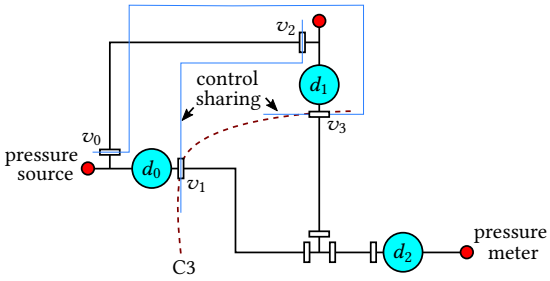

Figure 6: Valve sharing and its interfering with test and application execution.

4 VALVE SHARING IN TEST AND APPLICATION EXECUTION

In implementing single-source single-meter test for a multi-port biochip, new DFT channels and valves are added into the chip architecture. The newly added DFT valves also require control signals that switch them open or closed to regulate the fluid transportation. These control signals are also air pressure conducted to valves through control channels. If the newly added DFT valves are controlled independently, both control channels and external air pumps should be added, leading to an increase of the cost for manufacturing and running the chip. In this section, we propose a valve-sharing technique which eliminates the control overhead by sharing the control signals of DFT valves with those for original valves, while maintaining the performance of the chip.

4.1 Valve Sharing and its Validation

Figure 6 demonstrates how valves share controlling channels in a DFT architecture, where each new valve is connected to the control channel that is already existing to control another valve in the original chip. In this example, the DFT valve v_0 shares its control channel with v_3 and the DFT valve v_2 shares its control channel with v_1. Consequently, v_0 and v_3 always open and close simultaneously, and so do v_2 and v_1. This sharing scheme, however, interferes with the test procedure as well as the execution of the application. Therefore, we need to develop methods to validate a valve sharing scheme before exploring the design space of valve sharing.

Assume that v_1 and v_3 are kept closed during test to form a cut to separate the pressure source and the pressure meter. If v_3 cannot be closed properly due to defects, it is supposed that the pressure meter should be able to measure the leaked air pressure. However, with valve sharing shown in Figure 6, v_0 is also closed, masking the error supposed to be detected by this cut.

To validate whether a given valve sharing scheme is valid for the test procedure, each test vector, either a test path or a test cut, is applied and the output at the pressure meter is verified. For example, in the case in Figure 6, the validation can be performed by checking whether there is still a path from the pressure source to the meter after the test cut is activated and a valve, e.g., v_3, cannot be closed properly.

Similarly, valve sharing may also affect the execution of an application. Assume a transportation of fluid is performed from device d_0 to device d_1 through the upper path, so that v_0 must be opened. However, v_3 is also opened due to valve sharing, leading to a fluid leakage at d_1.

The validation of a valve sharing scheme during application execution is similar to the case for test. In a given schedule of an application, the occupation of all devices and channels at every moment are known. These snapshots of chip occupation are examined together with the

given valve sharing scheme to check whether all the operations can be performed correctly.

4.2 DFT with Valve Sharing using Particle Swarm Optimization (PSO)

As discussed above, DFT augmentation brings new channels and valves into the chip. Not only the locations of added channels and valves but also the control sharing scheme affect the test procedure and the execution of the application. In the proposed method, we deploy the Particle Swarm Optimization (PSO) algorithm to find valid DFT architectures that still maintain the efficiency of application execution.

Particle swarm optimization (PSO) is a stochastic population-based optimization algorithm [20]. Members of the population (swarm) are called particles. At the beginning of the algorithm, particles are randomly scattered into the whole search space. A particle p_i possesses a position x_i. A random velocity v_i is assigned to particle p_i and a cost function is applied for each particle to assess the quality of its current location. For each particle p_i, a local best position $p_{best,i}$ in its search history is tracked. For the whole swarm, a global best position g_{best} with the best assessment of all particles is also recorded. In each iteration, the new velocity and position of a particle p_i are updated as follows

$$v_i = \omega * v_i + c_1 * rand_1 * (x_i - p_{best,i}) + c_2 * rand_2 * (x_i - g_{best}) \quad (7)$$

$$x_i = x_i + v_i \quad (8)$$

where ω, c_1 and c_2 are constants to control the search speed. $rand_1$ and $rand_2$ are two random numbers to incorporate stochasticity into the search iterations. After each iteration, the performance at the new position of each particle and the global assessment are updated.

In the proposed method, the position of a particle consists of two parts of information: 1) how to add new channels to meet the single-source single-meter requirements; 2) how new valves share control channels with original valves. Therefore, we use two multidimensional vectors to denote these positions.

In Section 3, the optimization problem (5)–(6) may produce many viable solutions to meet the constraints and the objective. A vector $\vec{X}^a = [x_0, x_1, ..., x_n]$ is thus used to denote which edges in the connection grid are used to construct new channels. The definition of x_i is as follows.

$$x_i = \begin{cases} 1 & \text{if } e_i \text{ in the connection grid is used for DFT;} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where e_i represents an edge in the connection grid in Figure 5.

For valve sharing, assume that the number of valves for DFT is n_d and the number of valves in the original chip is n_v. A vector $\vec{X}^s$ is used to denote how DFT valves share control channels with original valves, defined as

$$\vec{X}^s = \{x_{i,j}\}, \qquad 0 \le i \le n_d, \quad 0 \le j \le n_v \quad (10)$$

$$x_{i,j} = \begin{cases} 1 & \text{if valve } v_i \text{ shares control channel with valve } v_j; \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

A PSO particle position is defined as the combination of the two vectors $\vec{X} = [\vec{X}^a, \vec{X}^s]$. The quality of a position is evaluated by two criteria. First, if the locations of DFT channels and valves as well as the corresponding control sharing scheme cannot pass the validation step described in Section 4.1, the quality of this position is set to ∞, meaning that the execution time of the application is so large that this position is invalid. If this position can pass the validation, its quality is set to the corresponding execution time of the application at this position.

In executing the PSO algorithm, valve sharing information can only be obtained after DFT channels and valves are created. Moreover, the validation of test and application execution can be performed after the valve sharing scheme is determined. Only after these steps, the execution time of the application can be evaluated. Consequently, multiple phases are required in each PSO iteration to generate new particle positions and to assess their quality.

In each PSO iteration, for a particle p_i with its position $\vec{X}_i = [\vec{X}_i^a, \vec{X}_i^s]$, the operations are as follows.

(1) $\vec{X}_i^a$ is updated according to the PSO search strategy, producing a new DFT configuration by solving (5)–(6).
(2) A sub-PSO algorithm is applied to find the best valve sharing scheme. First, randomized sub-particles with respect to different valve sharing schemes are generated. Their positions are set to $\vec{X}^s$. Since for each sub-particle, both new channels and valves as well as their sharing information are known, we can then assess the quality of these positions by validating them using the technique discussed in Section 4.1. After the sub-PSO algorithm is finished, a valve sharing with the minimum execution time of the application with respect to this sharing scheme is returned.
(3) By combining $\vec{X}_i^a$ and its best valve sharing scheme $\vec{X}_i^s$, we update the quality of the position $\vec{X}_i = [\vec{X}_i^a, \vec{X}_i^s]$ for p_i with the returned application execution time.
(4) After all particles find their new positions and their qualities are assessed, the local best position $p_{best,i}$ for particle p_i and the global best position g_{best} are updated accordingly. The iteration goes back to step (1) until the maximum allowed number of iterations is reached.

The PSO algorithm returns a DFT architecture together with its valve sharing scheme. The test vectors and application schedule considering valve sharing are also generated. The new architecture does not impose more control overhead, while facilitating the test process and maintaining the performance of application execution.

5 SIMULATION RESULTS

The proposed method was implemented in C++ and tested using a 3.20 GHz CPU with 8 GB memory. We demonstrate the results using three applications executed by three biochips. The information of these test cases are shown in Table 1, where IVD (In-Vitro Diagnostics), PID (Protein Interpolation Dilution) and CPA (Colorimetric Protein Assay) are real-world assays. The biochips used in the experiments were the IVD chip and RA30 chip from [6] and the mRNA chip from [21].

In Table 1, the results of DFT design of the chips above running the three applications are reported, where all chips have been modified to single-source single-meter architectures successfully. For each chip-application combination, the results are split into two rows, each of which contain three columns. In the first row, the number of added DFT valves, the number of the valves sharing control channels with existing valves and the runtime of the proposed method are shown. For all the combinations, the DFT valves have found a valid sharing mechanism. Compared with the number of original valves in the chips, the number of DFT valves is still in the acceptable range. The second row for an experiment combination reports the execution time of the corresponding application without DFT, with DFT but without PSO optimization for valve sharing and the final result of the whole framework. From this comparison, it can be seen that the introduction of DFT valves into a chip can initially prolong the execution time of an application significantly. With PSO optimization iterations, the execution time has been recovered back to nearly the same level of the original design, while in the case PID running on the RA30 chip the execution time has actually been reduced.

As shown in Table 1, the execution time of applications is slightly longer in many cases with DFT architectures than with the original chip. The main reason is that valve sharing blocks some transportation tasks

Table 1: Results of DFT Augmentation

Assay	IVD (12 op.)			PID (38 op.)			CPA (55 op.)		
IVD_chip (3 mixers, 2 detectors, 12 valves)	6	6	247	7	7	497	7	7	292
	270	580	310	840	1030	890	1220	1320	1320
RA30_chip (2 mixers, 3 detectors, 16 valves)	6	6	231	6	6	661	6	6	373
	270	440	280	950	1100	940	1140	1190	1190
mRNA_chip (3 mixers, 1 detectors, 28 valves)	4	4	612	4	4	733	4	4	892
	580	580	580	860	920	880	1640	1640	1640

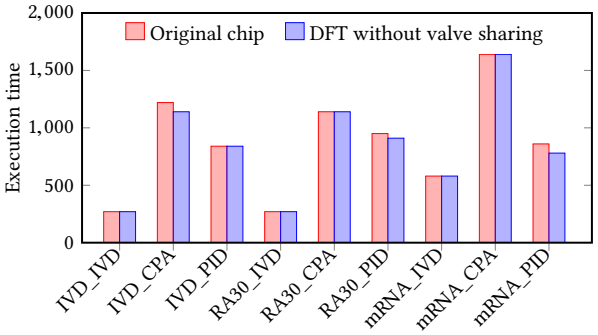

Figure 7: Comparison of execution time of applications using original chips and DFT architectures without valve sharing.

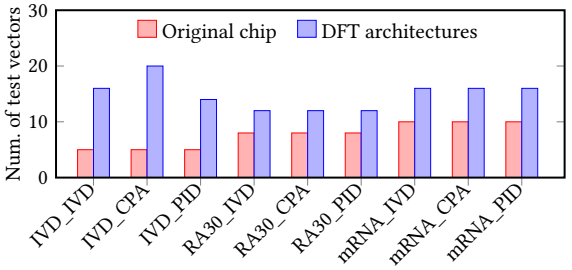

Figure 8: Comparison of numbers of test vectors in original chips and DFT architectures.

so that operations must wait for fluid samples. In the case that the DFT valves can have their own control channels, the execution time should be no larger than that in the original chip, because new resources added for DFT can also be used by applications. The results of this comparison are shown in Figure 7, where the execution performance is better in several cases.

In the DFT architectures, the number of pressure sources and meters have been reduced. Accordingly, the number of test vectors should increase because some test vectors become unavailable with this simplified test platform. Figure 8 compares the numbers of test vectors in the original chips and the DFT architectures. The larger number of test vectors leads to a relatively longer test time, which is still not a problem in today's biochemical laboratories.

In the process of PSO optimization, we used 5 particles for DFT valves and channels and 5 particles for valve sharing. The total number of iterations was set to 100. To show the converging trend of the PSO method, the execution times of applications with respect to the number of iterations are shown in Figure 9 for three chip-application combinations, where we can see that the results of PSO become stable at about 80 iterations, earlier than the number of iterations we used in the experiments.

6 CONCLUSION

In this paper, we have proposed a method to generate a biochip architecture with single-source single-meter testablity, so that the cost of the test platform can be reduced. The valves added for DFT share control channels with other existing valves so that no additional control ports are required. The performance of these chips in executing applications has also been maintained using particle swarm optimization.

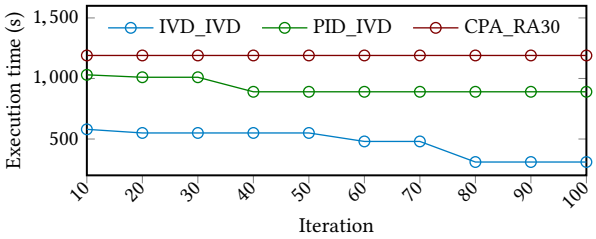

Figure 9: Execution time of applications during PSO iterations.

ACKNOWLEDGMENT

The work of B. Li and U. Schlichtmann was supported by Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE). The work of C. Liu was supported fully, and the work of K. Chakrabarty and T.-Y. Ho was supported in part, by the Technical University of Munich – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763.

REFERENCES

[1] J. M. Perkel, "Microfluidics: Binging new things to life science," *Science*, vol. 322, no. 5903, pp. 975–977, 2008.
[2] J. Melin and S. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 213–231, 2007.
[3] Illumina. [Online]. Available: http://www.illumina.com/
[4] Agilent. [Online]. Available: http://www.agilent.com/
[5] T.-M. Tseng, B. Li, U. Schlichtmann, and T.-Y. Ho, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 69–75, 2015.
[6] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, "Transport or store? synthesizing flow-based microfluidic biochips using distributed channel storage," in *Proc. Design Autom. Conf.*, 2017, pp. 49:1–49:6.
[7] K. Chakrabarty, R. B. Fair, and J. Zeng, "Design tools for digital microfluidic biochips: Toward functional diversification and more than moore," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 7, pp. 1001–1017, 2010.
[8] P. Pop, I. E. Araci, and K. Chakrabarty, "Continuous-flow biochips: Technology, physical-design methods, and testing," *IEEE Design & Test*, vol. 32, no. 6, pp. 8–19, 2015.
[9] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," in *Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embed. Sys.*, 2012, pp. 181–190.
[10] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. T. Lee, and T.-Y. Ho, "An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2014, pp. 141:1–141:6.
[11] K. Hu, T.-Y. Ho, and K. Chakrabarty, "Wash optimization and analysis for cross-contamination removal under physical constraints in flow-based microfluidic biochips," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 559–572, 2016.
[12] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 55–68, 2017.
[13] H. Yao, Q. Wang, Y. Ru, Y. Cai, and T.-Y. Ho, "Integrated flow-control codesign methodology for flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 60–68, 2015.
[14] H. Yao, T.-Y. Ho, and Y. Cai, "PACOR: practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2015, pp. 142:1–142:6.
[15] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1463–1475, 2014.
[16] C. Liu, B. Li, B. B. Bhattacharya, K. Chakrabarty, T.-Y. Ho, and U. Schlichtmann, "Testing microfluidic fully programmable valve arrays (FPVAs)," in *Proc. Design, Autom., and Test Europe Conf.*, 2017.
[17] K. Hu, T.-Y. Ho, and K. Chakrabarty, "Test generation and design-for-testability for flow-based mVLSI microfluidic biochips," in *Proc. VLSI Test Symp.*, 2014, pp. 97–102.
[18] K. S. Elvira, X. C. i Solvas, R. C. R. Wootton, and A. J. deMello, "The past, present and potential for microfluidic reactor technology in chemical synthesis," *Nature Chemistry*, no. 5, pp. 905–915, 2013.
[19] I. E. Araci and S. R. Quake, "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves," *Lab Chip*, vol. 12, pp. 2803–2806, 2012.
[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Networks*, 1995, pp. 1942–1948.
[21] J. S. Marcus, W. F. Anderson, and S. R. Quake, "Microfluidic single-cell mrna isolation and analysis," *Analytical Chemistry*, vol. 78, no. 9, pp. 3084–3089, 2006.