FT-MUX: A Fault-Tolerant Microfluidic Multiplexer Design

Mengchu Li
TU Munich
mengchu.li@tum.de

Jiahui Peng
TU Munich
jiahui.peng@tum.de

Tsun-Ming Tseng
TU Munich
tsun-ming.tseng@tum.de

Ulf Schlichtmann
TU Munich
ulf.schlichtmann@tum.de

***Abstract*—Continuous-flow microfluidic chips are multilayered miniaturized platforms to manipulate small volumes of fluids with valves. There are two types of channels on a chip: flow channels for the reaction of fluids, and control channels for the actuation of valves. Multiplexers (MUXes) are essential microfluidic components for individually addressing many flow channels with few control channels. As the integration scale of microfluidic chips increases, the reliability of MUXes becomes a critical concern, as a single defective control channel in a MUX will affect a large part of the flow channels addressed by the MUX. This paper formally analyzes and identifies the design rules for a MUX to tolerate n defective control channels, and model the fault-tolerant MUX (FT-MUX) design problem as a binary constant weight code problem to minimize resource overheads. We demonstrate that FT-MUX improves resource efficiency by up to hundreds of times compared to the conventional fault-tolerant design method. Besides, given no less than 10 control channels, FT-MUX tolerates at least one defective control channel and addresses even more flow channels with equal or fewer resources than a standard MUX. The advantages become more significant as the integration scale increases.**

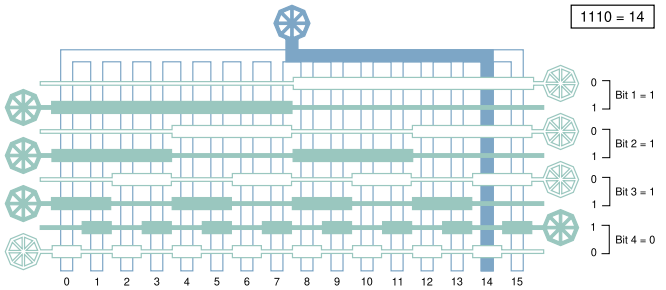

Fig. 1. A standard MUX to address 16 flow channels (colored in blue) with 8 control channels (colored in green). The pressure states in pressurized/ depressurized control channels are denoted as 1 and 0, respectively. Valves are widened control channel segments. A pressurized valve blocks the flow channel underneath. Every two control channels form a pair representing one bit of a flow channel index: pressure states $(0, 1)$ and $(1, 0)$ represent bit '1' and '0', respectively. E.g., configuring the control channels to pressure states $(0, 1)$ $(0, 1)$ $(0, 1)$ $(1, 0)$ constructs a unique flow path from the fluid inlet to flow channel 14 (1110). Figure adapted from [2].

I. Introduction

Microfluidic technology enables the manufacturing of miniaturized devices, also known as microfluidic chips, that manipulate fluids at the nanoliter scale or below, exhibiting advantages in automation, rapid execution, better sensitivity, less waste, etc. In particular, the fabrication of valves using multilayer soft lithography [1] allows for the creation of microfluidic large-scale integration (mLSI) chips [2]. Such a chip consists of a flow layer (integrated) with flow channels for the storage and transportation of fluid samples and reagents, and a control layer (integrated) with control channels for the actuation of valves to manipulate the fluids. Nowadays, it is common to have hundreds to thousands of flow channels on a single small chip for high-throughput parallel applications such as drug screening [3], cell-isolation [4], cell-culture [5], cell-analysis [6], etc. To independently address many flow channels with an affordable number of control inputs, microfluidic multiplexers (MUXes) have become indispensable for large-scale microfluidic applications [7].

A standard microfluidic multiplexer (MUX) [8] can independently address up to n flow channels with $2\log_2(n)$ control channels, as shown in Fig. 1. To that end, each control channel is responsible for the pressure supply to valves on at least $\lfloor \frac{n}{2} \rfloor$ flow channels. Compared to flow channels, control channels have smaller feature sizes and are thus more prone to defects [9]. If a control channel in a standard MUX is defective, at least half of the flow channels addressed by the MUX will be affected, leading to waste of samples and reagents, contamination of products, or even disposal of the whole chip. As the integration scale of microfluidic chips increases, the reliability of MUXes becomes a critical concern [10].

Efforts have been made in the past to improve the reliability of MUXes from two aspects:

1) Reduce the likelihood of defects. Q. Wang et al. proposed to optimize the switching order of valves to reduce the switching frequency of a MUX [11], and S. Liang et al. proposed a novel MUX design with certain valves and channels merged together [12]. However, as the number of control channels in the MUX increases, defects are almost inevitable, as reported in [12]. Since the proposed methods do not tolerate faults, once a defect happens, many flow channels will be affected, leading to significant waste.
2) Tolerate defects by introducing backups. W. Huang et al. proposed an approach to introduce redundant components considering a given set of fault scenarios [13], and Y. Zhu et al. proposed an approach to synthesize application-specific MUXes with duplicated valves and backup paths to actuate the valves [14]. However, the performances of both approaches heavily depend on the given applications and are thus, in general, not predictable.

Besides, state-of-the-art methods mostly assume that defects happen at individual valves without influencing other valves

along the same control channel, which is not the case. Specifically, microfluidic chips suffer two major types of defects: blockage and leakage [15]. Blockage means a channel is disconnected such that pressure cannot pass through, and the risk of blockage inversely correlates with the feature size. Since valves are widened segments of control channels, blockage defects are more likely at the thinner parts of a control channel instead of valves. When a control channel is blocked, all valves along the channel cannot be properly pressurized. On the other hand, leakage means two control channels are unintentionally connected such that both channels share the same pressure states. When a control channel leaks to another, pressurizing/depressurizing one channel will also inevitably close/open all valves in the other channel. As conclusion, a defective valve in the MUX implies the malfunctions of all valves along a defective control channel. Thus, a fault-tolerant MUX design should target defective control channels rather than defective valves.

In this paper, we propose a rule for designing a fault-tolerant microfluidic multiplexer (FT-MUX) to tolerate n defective control channels with minimal resource overheads. We further model the FT-MUX design problem as a binary constant weight code problem, and proposes a graph model to solve it as an independent vertex set problem. We demonstrate that FT-MUX improves resource efficiency by up to hundreds of times compared to the conventional fault-tolerant design method. Besides, given no less than 10 control channels, FT-MUX tolerates at least one defective control channel and addresses even more flow channels with equal or fewer resources than a standard MUX. The performance advantages become more significant as the integration scale increases, showing that FT-MUX provides an efficient and reliable solution for fluid multiplexing in large-scale microfluidic applications.

II. Background

A. General Design Rules for MUXes

A MUX is a matrix of binary valve patterns on overlapped control and flow channels. Given a MUX, we consider every control channel as a row of the matrix, with the topmost control channel as the first row, and every flow channel as a column of the matrix, with the leftmost flow channel as the first column. We denote the entry in the i^{th} row and j^{th} column of the matrix as '0' or '1', depending on whether there is no or one valve at the intersection of the i^{th} control channel and the j^{th} flow channel, respectively. Thus, the matrix column representing flow channel 14 of the standard MUX in Fig. 1 can be read as 10101001 (see captions of Fig. 1), referred to as the ***code*** of flow channel 14.

To construct a flow path from a universal fluid inlet to a flow channel f, one needs to pressurize any control channel that *does not have* a valve on its intersection with f, and depressurize any control channel that *has* a valve on its intersection with f. In other words, to address a flow channel f, if the j^{th} bit of the code of f is '0', the j^{th} control channel should be pressurized, and if the j'^{th} bit of the code of f is '1', the j'^{th} control channel should be depressurized. Thus, flow channel 14 in Fig. 1 can be addressed by setting the pressure states in the control channels to $\overline{1}0\overline{1}0\overline{1}0\overline{0}\overline{1}$, i.e., 01010110, referred to as the ***control pattern*** of flow channel 14, where '1' or '0' in the i^{th} bit represents that the i^{th} control channel needs to be *pressurized* or *depressurized*, respectively.

To ensure that every flow channel can be independently addressed by the MUX, i.e., fluids can flow from a universal fluid inlet to an arbitrary flow channel without contaminating other flow channels, the design of a MUX must obey the following rule:

Rule 1: The control pattern of an arbitrary flow channel f pressurizes at least one valve on every flow channel other than f.

The standard MUX obeys the rule by assigning a unique $\lceil \log_2(n) \rceil$-bit *index* [1] to each of n flow channels and representing each bit of the index with binary strings 01 or 10 to construct the *code* of that flow channel. Thus, if the indices of two flow channels f and f' differ in the j^{th} bit, the valve on the intersection between the j^{th} pair of control channels and f must have a different pressure state than the valve on the intersection between the j^{th} pair of control channels and f'. Since the indices of any two flow channels must differ by at least one bit, the control pattern of f pressurizes at least one valve on f'. E.g., the index of flow channel 15 (1111) differs from flow channel 14 (1110) in the last bit, and therefore, the control pattern of flow channel 14 pressurizes the last valve on flow channel 15, as shown in Fig. 1.

S. Liang et al. further proposed not to pair up the control channels and concluded the following rule to fully exploit the binary coding capacity of MUXes [12]:

Rule 2: The maximal coding capacity of a MUX consisting of n control channels is $C^n_{\lfloor \frac{n}{2} \rfloor}$, which can be achieved by enumerating all n-bit binary strings with $\lfloor \frac{n}{2} \rfloor$ '1'-bits.

E.g., the maximal coding capacity of a MUX consisting of 4 control channels is $C^4_2 = \frac{4!}{2! \cdot (4-2)!} = 6$, which can be achieved by enumerating all 4-bit binary strings with two '1'-bits, i.e., 0011, 0101, 1001, 1010, 1100, 0110.

B. Control-Channel Defects in MUXes

Microfluidic chips suffer two major types of defects: *blockage* and *leakage*. Valves on a blocked control channel cannot be pressurized and thus lose their functions, and valves on a pair of leaky control channels cannot be independently pressurized but have to share the same pressure states. A defective control channel in a standard MUX will affect $50\% - 100\%$ of the flow channels addressed by the MUX, depending on the type and position of the defect.

Fig. 2(a) shows the working mechanism of a standard MUX that addresses 8 flow channels with 6 control channels.

- If there is a blockage defect, half of the flow channels addressed by the MUX will be affected. Specifically, suppose that the j^{th} control channel is blocked. Any flow channel with a '1' in the j^{th} bit of its code will be affected. For example, the first control channel in the

[1] Note that "index" is a term only used in standard MUXes.

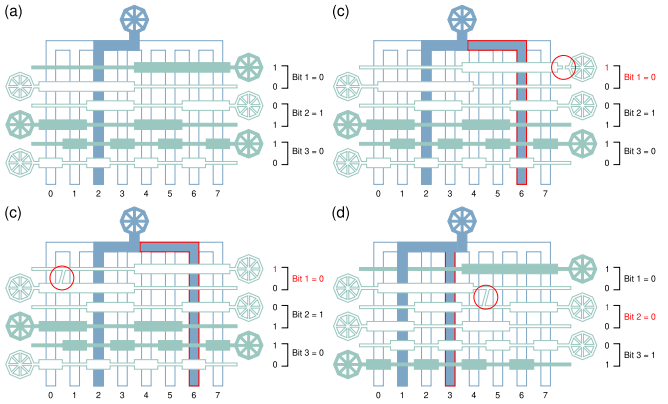

Fig. 2. A standard MUX (a) without defects; (b) with one blockage defect; (c) with a leakage defect between two control channels representing the same bit of the flow channel index; (d) with a leakage defect between two control channels representing different bits of the flow channel index.

standard MUX shown in Fig. 2(b) is blocked. As a result, flow channels, the indices of which start with a '0', i.e., the codes of which start with 10, will be affected. Such channels include flow channels 0 (000), 1 (001), 2 (010), and 3 (011). E.g., fluids cannot flow from the fluid inlet to flow channel 2 without contaminating flow channel 6 since the control pattern of flow channel 2 can no longer pressurize any valve on control channel 6.

- If there is a leakage defect between a pair of control channels representing the same bit of the flow channel index, all flow channels addressed by the MUX will be affected. Specifically, suppose that the j^{th} pair of control channels leak with each other. Pressurizing either of them will block all flow channels addressed by the MUX, and depressurizing both of them will prevent two flow channels, the binary indices of which differ only in the j^{th} bit, from being addressed independently. E.g., the first pair of control channels in the standard MUX shown in Fig. 2(c) leak with each other. As a result, flow channels 2 (010) and 6 (110), the binary indices of which only differ in the first bit, cannot be independently addressed.
- If there is a leakage defect between a pair of control channels representing different bits of the flow channel index, half of the flow channels addressed by the MUX will be affected. Specifically, suppose that the two leaky control channels represent the j_1^{th} and j_2^{th} bits of the flow channel index. Any flow channel with the same value in the j_1^{th} and j_2^{th} bits of its index will be affected. For example, the second and third control channels in the standard MUX shown in Fig. 2(d) leak with each other. As a result, flow channels, the first two bits of the indices of which are 00 or 11, will be affected. Such channels include flow channels 0 (000), 1 (001), 6 (110) and 7 (111). E.g., fluids cannot flow from the fluid inlet to flow channel 1 without contaminating flow channel 3 since the control pattern of flow channel 1 can no longer pressurize any valve on control channel 3.

Similarly, a defective control channel in the novel MUX design proposed in [12], referred to as CoMUX, will affect $50\% - 100\%$ of the flow channels addressed by the MUX, leading to the complete waste of the reagents, samples, and chip area. Besides, the risk of defects increases with the integration scale of the chip. More damage is expected if there are more defective control channels in the MUX. Thus, the design of a fault-tolerant MUX with minimal resource overhead is essential.

III. Problem Formulation

Input:

- The number of control channels in the MUX, denoted as n.
- The number of control channel defects that should be tolerated by the MUX, denoted as k.

Output:

- A fault-tolerant MUX design consisting of n control channels to independently address every flow channel in the MUX regardless of k control channel defects.

Objective:

- Maximize the number of flow channels addressed by the fault-tolerant MUX.

IV. Method

A. Designs Rules for Fault-Tolerant MUXes

For convenience, we denote the code of a flow channel f as c_f. As mentioned in Section II-A, a MUX design must obey Rule 1:

Rule 1: The control pattern of an arbitrary flow channel f pressurizes at least one valve on every flow channel other than f.

To develop the design rule for a fault-tolerant MUX, we first transform Rule 1 to the following equivalent statement:

Rule 1^*: For any two flow channels f and f', c_f and c'_f must differ by at least two bits, among which there are at least one bit at which c_f takes '1' and c'_f takes '0' and one bit at which c_f takes '0' and c'_f takes '1'.

Proof of the equivalence:

- '$\Rightarrow$': Suppose that the control pattern of f pressurizes the valve on the intersection between the j^{th} control channel and f'. The j^{th} bits of c_f and $c_{f'}$ must be 0 and 1, respectively. Similarly, suppose that the control pattern of f' pressurizes the valve on the intersection between the j'^{th} control channel and f. The j'^{th} bits of c_f and $c_{f'}$ must be 1 and 0, respectively. Thus, Rule 1^* is satisfied.
- '$\Leftarrow$': Suppose that the j^{th} and j'^{th} bits of c_f and $c_{f'}$ are 01 and 10, respectively. The control pattern of f pressurizes the valve on the intersection between the j^{th} control channel and f'. Thus, Rule 1 is satisfied. □

Next, we propose the following rule for designing a MUX that tolerates one blockage or leakage defect:

Rule 3: For any two flow channels f and f', c_f and c'_f must differ by at least four bits, among which there are at least two

bits at which c_f takes '1' and c'_f takes '0' and two bits at which c_f takes '0' and c'_f takes '1'.

In the following, we prove that if a MUX satisfies Rule 3, it satisfies Rule 1^* even when there is one blockage or leakage defect.

- Suppose that there is a blockage defect on the j^{th} control channel. Valves along the j^{th} control channel cannot be pressurized and should thus be ignored, i.e., the j^{th} bits of all codes should be regarded as 0. In this case, at least three bits of c_f and c'_f remain different. Thus, Rule 1^* is satisfied.
- Suppose that there is a leakage defect between the j^{th} and j'^{th} control channels. Pressurizing one channel will close all valves along the other channel. In other words, to address a certain flow channel, if either the j^{th} or the j'^{th} control channel is required to be depressurized, both the j^{th} and the j'^{th} control channels must be depressurized. Thus, if either the j^{th} or the j'^{th} bit of a code is '1', the other bit of the code should also be regarded as '1'. We denote the j^{th} and j'^{th} bits of a code c_f as s_f, where 's' stands for 'subset', and the j^{th} and j'^{th} bits of another code $c_{f'}$ as $s_{f'}$, with $s_f, s_{f'} \in \{00, 01, 10, 11\}$:
  - If $s_f, s_{f'} \in \{00, 11\}$, the leakage does not affect c_f and c'_f. Thus, Rule 1^* is satisfied.
  - If $s_f \in \{10, 01\}$ and $s_{f'} = 00$ or vice versa, due to the leakage, s_f should be regarded as 11. Thus, one more bit of c_f and c'_f becomes different, and Rule 1^* is satisfied.
  - If $s_f \in \{10, 01\}$ and $s_{f'} = 11$ or vice versa, due to the leakage, s_f should be regarded as 11. Thus, at least three bits of c_f and c'_f remain different, and Rule 1^* is satisfied.
  - If $s_f = 01$ and $s_{f'} = 10$ or vice versa, due to the leakage, both s_f and $s_{f'}$ should be regarded as 11. Thus, at least two bits of c_f and c'_f remain different, and among the remaining different bits, there are still at least one bit at which c_f takes '1' and c'_f takes '0' and one bit at which c_f takes '0' and c'_f takes '1', i.e., Rule 1^* is satisfied.
  - If $s_f = s_{f'}$, the leakage does not affect the bit differences between c_f and c'_f. Thus, Rule 1^* is satisfied. □

We further generalize Rule 3 as follows for designing a MUX that tolerates k blockage or leakage defects:

Rule 4: For any two flow channels f and f', c_f and c'_f must differ by at least $2(k+1)$ bits, among which there are at least $k+1$ bits at which c_f takes '1' and c'_f takes '0' and $k+1$ bits at which c_f takes '0' and c'_f takes '1'.

The proof can be easily derived using the same strategy as above.

B. Binary Constant Weight Code Model

To maximize the number of flow channels addressed by the fault-tolerant MUX, we model the fault-tolerant MUX design problem as a binary constant weight code problem.

In coding theory, a binary constant weight code with parameters n, w, d is a set $\mathcal{C}$ of binary words of length n and Hamming weight w such that the Hamming distance between any two binary words in $\mathcal{C}$ is at least d [16]. Specifically, a binary word of length n is a string of n bits; the Hamming weight of a binary word is the number of '1'-bits in the binary word; and the Hamming distance between two binary words of equal length is the number of bits at which the two words take different values.

According to Rule 2, the largest set of codes of flow channels in a MUX consisting of n control channels, denoted as $\mathcal{M}_n$, can be achieved by enumerating all n-bit binary strings with $\lfloor \frac{n}{2} \rfloor$ '1'-bits. Since a fault-tolerant MUX is a MUX with additional constraints on the selection of codes, the largest set of codes of the flow channels in a fault-tolerant MUX consisting of n control channels should be a subset of $\mathcal{M}_n$. Thus, we derive the following statement:

To maximize the coding capacity of a fault-tolerant MUX consisting of n control channels, the code of every flow channel in the MUX should be a binary word of length n and Hamming weight $\lfloor \frac{n}{2} \rfloor$.

We also observed that for two binary words w_1 and w_2 of the same length and Hamming weight, if the Hamming distance between w_1 and w_2 is $2k$, there must be exactly k bits at which w_1 takes '1' and w_2 takes '0' and k bits at which w_1 takes '0' and w_2 takes '1'. Thus, given the same length and Hamming weight of the codes of all flow channels in a MUX, Rule 4 can be simplified as:

Rule 4^*: For any two flow channels f and f', the Hamming distance between c_f and c'_f must be larger than or equal to $2(k+1)$.

Therefore, the problem described in Section III can be modeled as the following binary constant weight code problem:

Finding a largest possible binary constant weight code with length n, weight $\lfloor \frac{n}{2} \rfloor$, and Hamming distance $2(k+1)$.

The binary constant weight code problem has been extensively studied in the field of information theory [17], [18]. Some upper bounds and lower bounds of the maximum size of a binary constant weight code can be computed [19], but it is generally difficult to find the largest possible code for most parameters. To approximate the theoretical upper bounds, we propose a graph model and transform the binary constant weight code problem into an independent vertex set problem.

Given n control channels and k to be tolerated defects, we build a graph $G = (V, E)$, in which each vertex $v \in V$ represents a binary word of length n and Hamming weight $\lfloor \frac{n}{2} \rfloor$. If the Hamming distance between the binary words represented by any two vertices v_1 and v_2 is smaller than $2(k+1)$, we add an edge $(v_1, v_2) \in E$. Therefore, any independent vertex set of the graph represents a binary constant weight code with length n, weight $\lfloor \frac{n}{2} \rfloor$, and Hamming distance $2(k+1)$. The problem of finding the largest possible binary constant weight code is thus equivalent to finding the maximum independent vertex set in graph G.

The maximum independent vertex set problem is a classic

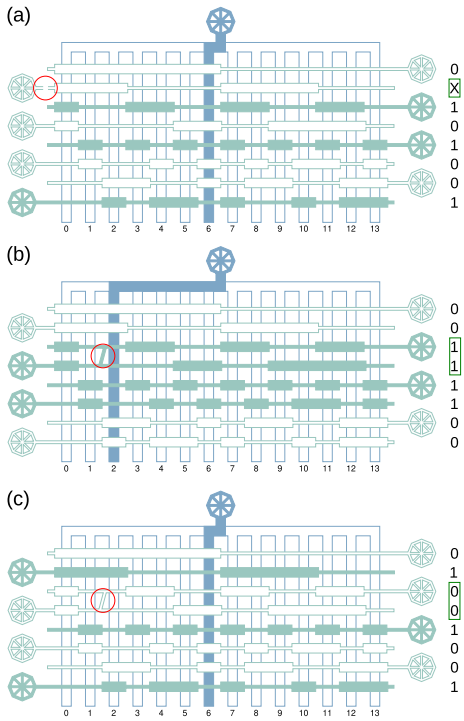

Fig. 3. (a) Control pattern in case of a blockage defect. (b)(c) Control patterns in case of a leakage defect.

NP-hard problem, for which there are numerous studies and algorithms. In this work, we solve the problem with a simple mixed integer linear programming model:

For every vertex $v \in V$, we introduce a binary variable b_v to represent whether v is in the target independent vertex set. We then introduce the following constraint for each edge $(v_1, v_2) \in E$:

$$b_{v_1} + b_{v_2} \leq 1,$$

and set the objective function as:

$$\textit{Maximize:} \sum_{v \in V} b_v.$$

C. Control Patterns of Fault Tolerant MUXes

Based on a binary constant weight code $\mathcal{C}$ with length n, weight $\lfloor \frac{n}{2} \rfloor$, and Hamming distance $2(k+1)$, we can design a MUX that addresses $|\mathcal{C}|$ flow channels with n control channels tolerating k defects, where $|\mathcal{C}|$ denotes the cardinality of set $\mathcal{C}$.

For example, Fig. 3 shows a MUX designed based on the largest binary constant weight code with length 8, weight 4, and Hamming distance 4, which addresses 14 flow channels with 8 control channels tolerating one defect. The control pattern to address an arbitrary flow channel with the fault-tolerant MUX depends on the code of the flow channel and the type of defect.

- If there is a blockage defect on any control channel, the control pattern to address a flow channel remains the complement of the code of that flow channel. E.g. the second control channel of the fault-tolerant MUX in Fig. 3(a) suffers a blockage defect. Nevertheless, the control pattern to address flow channel 6 with code 10010110 remains $\bar{1}\bar{0}\bar{0}\bar{1}\bar{0}\bar{1}\bar{1}\bar{0}$, i.e. 01101001. In fact, since the second control channel cannot be properly pressurized, we can also denote the second bit of the control pattern as 'X', representing a don't care term.
- If there is a leakage defect between the j^{th} and j'^{th} control channels, e.g. between the 3rd and the 4th control channels, as shown in Fig. 3(b) and (c), there are two cases:
  - If the j^{th} and j'^{th} bits of the code of the to-be-addressed flow channel take the same value, i.e. 00 or 11, the control pattern remains the complement of the code of that channel. E.g. as shown in Fig. 3(b), the control pattern to address flow channel 2 with code 11000011 remains $\bar{1}\bar{1}\bar{0}\bar{0}\bar{0}\bar{0}\bar{1}\bar{1}$, i.e. 00111100, since the 3rd and the 4th bits of the code of channel 2 are both '0'.
  - If the j^{th} and j'^{th} bits of the code of the to-be-addressed flow channel take different values, i.e. 01 or 10, we change the j^{th} and j'^{th} bits of the control pattern to 00, and keep other bits of the control pattern the complements of the code of that channel. E.g. as shown in Fig. 3(c), the control pattern to address flow channel 6 with code 10010110 becomes $\bar{1}\bar{0}00\bar{1}\bar{1}\bar{0}$, i.e. 01001001, since the 3rd and the 4th bits of the code of channel 6 are '0' and '1', respectively.

V. Results

We compare our fault-tolerant MUX design, abbreviated as FT-MUX, with two MUX designs: the mainstream design, referred to as the standard MUX [8] and a novel design, referred to as CoMUX [12]. Since the standard MUX and CoMUX do not tolerate defects, we apply the conventional fault-tolerant design method [13] to duplicate the control channels and valves in these MUXes. Specifically, since it is in general not possible to predict the position of defects, to tolerate one or two defects, all control channels in a standard MUX/CoMUX need to be doubled or tripled, respectively. We denote MUXes that tolerate one or two defects by adding (I) or (II) next to their names, respectively.

To achieve the optimal code for our FT-MUX, we run the mixed integer linear programming model on a computer with two Xeon Gold 6126 2.6 GHz processors. We achieved the optimal solutions, i.e. the largest possible codes, for FT-MUX(I) and FT-MUX(II) with up to 12 control channels within a few seconds. Given more than 12 control channels, the model cannot guarantee the optimality of the solution. In those cases, we adopt the constant weight code table published in [20]. The results of the comparison are shown in Fig. 4. Based on the comparison, we make the following conclusions:

- FT-MUX greatly improved the resource efficiency compared to the conventional fault-tolerant design method. In particular, the efficiency advantages increase with the

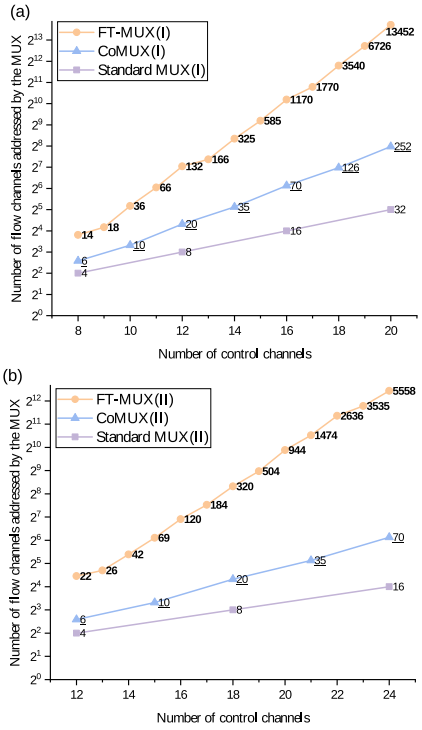

Fig. 4. Comparison of resource efficiency among (a) MUXes that tolerate one defect; (b) MUXes that tolerate two defects.

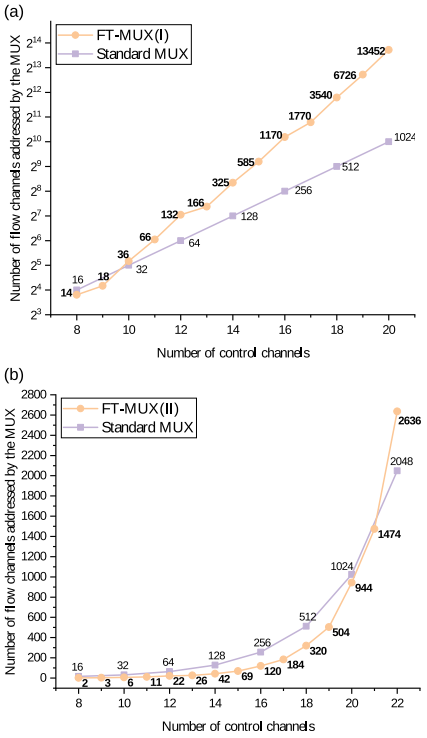

Fig. 5. Comparison of resource efficiency between (a) FT-MUXes that tolerate one defect and standard MUXes that tolerate no defect; (b) FT-MUXes that tolerate two defects and standard MUXes that tolerate no defect.

integration scale. Given 12 control channels, FT-MUX(I) addresses 6 and 16 times more flow channels than CoMUX(I) and standard MUX(I), respectively; and given 20 control channels, FT-MUX(I) addresses 53 and 420 times more flow channels than CoMUX(I) and standard MUX(I), respectively.

- FT-MUX is the only reasonable MUX design to tolerate more than one defect. In case two defects need to be tolerated, with up to 18 control channels, CoMUX(II) and standard MUX(II) require a similar or even larger number of control channels than the to-be-addressed flow channels, making the multiplexing meaningless. In the meanwhile, FT-MUX(II) can still reliably address many flow channels with only a few control channels.

Besides, given 24 control channels, FT-MUX(III) can tolerate three defects while addressing 2576 flow channels; and given 32 control channels, FT-MUX(IV) can tolerate four defects while addressing 3038 flow channels.

To demonstrate the efficiency of FT-MUXes, we also compare them to the original design of the standard MUX without fault-tolerant features. Fig. 5 shows the results of the comparison. Specifically, given no less than 10 control channels, FT-MUX(I) tolerates one defect while addressing more flow channels than a standard MUX that tolerates no defect; and given no less than 22 control channels, FT-MUX(II) tolerates two defects while addressing more flow channels than a standard MUX that tolerates no defect. In other words, FT-MUX significantly outperforms the standard MUX, not only in reliability but also in resource efficiency.

It is worth mentioning that the original design of CoMUX offers the largest coding capacity, i.e., addresses the most flow channels with a given number of control channels. However, as mentioned in Section II-B, a single control channel defect in CoMUX will affect 33% - 50% of the flow channels addressed by the MUX, leading to the waste of up to half of the reagents, samples, and chip area. As the risk of defects increases with the integration scale, the larger a microfluidic design is, the more fault tolerance should be considered. For large-scale designs, FT-MUX offers a more reliable and, thus, better solution than CoMUX.

VI. Conclusion

Fluid multiplexing is one of the most important operations on microfluidic chips. The reliability of fluid multiplexers, or MUXes, has thus been drawing more and more attention in recent years. This paper proposes the design rules for a fault-tolerant MUX (FT-MUX) for the first time and models the FT-MUX design problem as a constant weight code problem to minimize resource overheads. We demonstrated that FT-MUX significantly outperforms the state-of-the-art MUX designs and fault-tolerant design methods. As the integration scale of microfluidic chips increases, FT-MUX enables an efficient and reliable solution to execute microfluidic applications.

Acknowledgment

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, No. 515003344).

REFERENCES

[1] Marc A Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, and Stephen R Quake. Monolithic microfabricated valves and pumps by multilayer soft lithography. *science*, 288(5463):113–116, 2000.

[2] Jessica Melin and Stephen R Quake. Microfluidic large-scale integration: the evolution of design rules for biological automation. *Annu. Rev. Biophys. Biomol. Struct.*, 36:213–231, 2007.

[3] Zhanhui Wang, Min-Cheol Kim, Manuel Marquez, and Todd Thorsen. High-density microfluidic arrays for cell cytotoxicity analysis. *Lab on a Chip*, 7(6):740–745, 2007.

[4] Klaus Eyer, Phillip Kuhn, Conni Hanke, and Petra S Dittrich. A microchamber array for single cell isolation and analysis of intracellular biomolecules. *Lab on a Chip*, 12(4):765–772, 2012.

[5] Nina Compera, Scott Atwell, Johannes Wirth, Christine von Törne, Stefanie M Hauck, and Matthias Meier. Adipose microtissue-on-chip: a 3d cell culture platform for differentiation, stimulation, and proteomic analysis of human adipocytes. *Lab on a Chip*, 22(17):3172–3186, 2022.

[6] Mohamed Ibrahim, Krishnendu Chakrabarty, and Ulf Schlichtmann. Cosyn: Efficient single-cell analysis using a hybrid microfluidic platform. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1673–1678. IEEE, 2017.

[7] Tsun-Ming Tseng, Mengchu Li, Daniel Nestor Freitas, Amy Mongersun, Ismail Emre Araci, Tsung-Yi Ho, and Ulf Schlichtmann. Columba s: A scalable co-layout design automation tool for microfluidic large-scale integration. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[8] Todd Thorsen, Sebastian J Maerkl, and Stephen R Quake. Microfluidic large-scale integration. *Science*, 298(5593):580–584, 2002.

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

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

[11] Qin Wang, Shiliang Zuo, Hailong Yao, Tsung-Yi Ho, Bing Li, Ulf Schlichtmann, and Yici Cai. Hamming-distance-based valve-switching optimization for control-layer multiplexing in flow-based microfluidic biochips. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 524–529. IEEE, 2017.

[12] Siyuan Liang, Mengchu Li, Tsun-Ming Tseng, Ulf Schlichtmann, and Tsung-Yi Ho. Comux: Combinatorial-coding-based high-performance microfluidic control multiplexer design. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[13] Wei-Lun Huang, Ankur Gupta, Sudip Roy, Tsung-Yi Ho, and Paul Pop. Fast architecture-level synthesis of fault-tolerant flow-based microfluidic biochips. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1667–1672. IEEE, 2017.

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

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

[16] Kenneth S Suslick. Encyclopedia of physical science and technology. *Sonoluminescence and sonochemistry, 3rd edn. Elsevier Science Ltd, Massachusetts*, pages 1–20, 2001.

[17] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith. A new table of constant weight codes. *IEEE Trans. Inf. Theory*, 36:1334–1380, 1990.

[18] Derek H Smith, Lesley A Hughes, and Stephanie Perkins. A new table of constant weight codes of length greater than 28. *the electronic journal of combinatorics*, 13(1):A2, 2006.

[19] Selmer Johnson. A new upper bound for error-correcting codes. *IRE Transactions on Information Theory*, 8(3):203–207, 1962.

[20] Bounds for binary constant weight codes. https://www.win.tue.nl/ aeb/codes/Andw.html. Accessed: 2024-07-05.