CoMUX: Combinatorial-Coding-Based High-Performance Microfluidic Control Multiplexer Design

Siyuan Liang
The Chinese University of Hong Kong
Shatin, Hong Kong, China
siyuan.liang@link.cuhk.edu.hk

Mengchu Li
Technical University of Munich
Munich, Germany
mengchu.li@tum.de

Tsun-Ming Tseng
Technical University of Munich
Munich, Germany
tsun-ming.tseng@tum.de

Ulf Schlichtmann
Technical University of Munich
Munich, Germany
ulf.schlichtmann@tum.de

Tsung-Yi Ho
The Chinese University of Hong Kong
Shatin, Hong Kong, China
tyho@cse.cuhk.edu.hk

ABSTRACT

Flow-based microfluidic chips are one of the most promising platforms for biochemical experiments. Transportation channels and operation devices inside these chips are controlled by microvalves, which are driven by external pressure sources. As the complexity of experiments on these chips keeps increasing, control multiplexers (MUXes) become necessary for the actuation of the enormous number of valves. However, current binary-coding-based MUXes do not take full advantage of the coding capacity and suffer from the reliability problem caused by the high control channel density. In this work, we propose a novel MUX coding strategy, named Combinatorial Coding, along with an algorithm to synthesize combinatorial-coding-based MUXes (CoMUXes) of arbitrary sizes with the proven maximum coding capacity. Moreover, we develop a simplification method to reduce the number of valves and control channels in CoMUXes and thus improve their reliability. We compare CoMUX with the state-of-the-art MUXes under different control demands with up to 10×2^{13} independent control channels. Experiments show that CoMUXes can reliably control more independent control channels with fewer resources. For example, when the number of the to-be-controlled control channels is up to 10×2^{13}, compared to a state-of-the-art MUX, the optimized CoMUX reduces the number of required flow channels by 44% and the number of valves by 90%.

1 INTRODUCTION

Flow-based microfluidic biochips have become an increasingly attractive platform for biochemical experiments during the past decades [1, 2]. These coin-sized chips can incorporate many miniaturized devices to carry out complex operations that are traditionally performed in cumbersome laboratory instruments [3], and therefore enjoy many advantages including small consumption of reagents, increased automation degree [4], and reduced manufacturing costs [5]. Such a chip consists of two layers: a flow layer containing the flow channels and a control layer containing the control channels. On-chip devices such as mixers and detectors are connected by flow channels to transport the fluids. The fluid transportation is controlled by microvalves [6, 7], which are tiny switches built at the intersections of the flow channels and the control channels. Figure 1(a) shows the 3D schematic of a valve and the corresponding channel connections. In order to control the fluid transportation, we can transport pressure from the control port through the control channel to the valve. When the pressure is low, fluids in the flow channel can safely pass the valve. In that case, we say that the valve is *open*. By contrast, when the pressure is high, the inflated control channel will squeeze the flow channel and block the fluid movement. In that case, we say that the valve is *closed*. Figure 1(b) shows the moment when a valve is closing and separates the fluids into two parts.

A main challenge in the development of microfluidic biochips is to reduce the dependency on external pneumatic controllers. Specifically, fluids and pressure are transported between on-chip and off-chip components via flow and control ports. A flow/control port is a punch hole on the chip. Each port consumes remarkable chip area and requires an external pneumatic controller. As the complexity of experiments executed on microfluidic biochips increases, the number of valves increases so significantly that it becomes impractical to assign an independent control port to every valve. Thus, using multiplexers (MUXes) to control the valves has become a necessary choice [8, 9].

The structure of the MUX was first proposed by Thorsen et al. [10] to address 2^N flow channels individually with $2N$ control ports based on binary coding. L. Fidalgo and S. Maerkl further proposed to use flow channels to address control channels and thereby enabled individual control of 2^N valves with $2N$ flow ports [11]. Q. Wang et al. proposed a method to optimize the switching order of valves to enhance the reliability of MUXes [12]. Y. Zhu et al. proposed a method named multi-control to synthesize application-specific MUXes that address some control channels simultaneously. The spared coding capacity is then used to introduce backup control channels to improve the tolerance of control channel defects [13].

Despite the advances, current binary-coding based MUXes have not taken full advantage of the coding capacity. As a result, they require more channels and ports than necessary, which leads to area

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549353

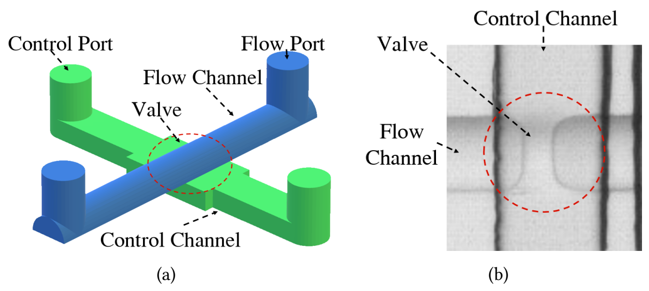


Figure 1: (a) Three-dimensional schematic diagram of a valve. (b) A closing valve separating the fluid in the flow channel.

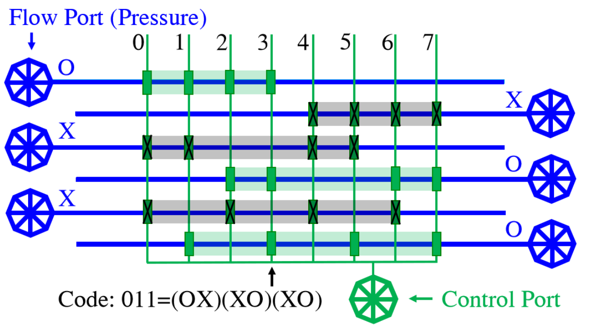


Figure 2: Example of a classic MUX that uses 6 flow channels to individually address 8 control channels.

overhead and an excessive chip-to-world interface. For example, in middle-scale designs, a MUX can easily consume about half of the chip area [11, 14].

Moreover, control channels on microfluidic chips may have blockage and leakage defects [15, 16]. The valve-switching method [12] only improves the reliability of valves but cannot improve the reliability of control channels; the multi-control method [13] introduces backup channels which decrease the risk of blockage, but the increased number of channels results in a higher risk of leakage.

In this paper, we propose the combinatorial coding strategy and a new MUX design named combinatorial-coding-based multiplexer (CoMUX). The CoMUX achieves the provably maximum coding capacity and is more robust against valve and channel defects compared to classic MUXes.

The rest of the paper is organized as follows: In section 2, we introduce the working mechanism of a classic MUX and common defects on microfluidic chips. In section 3, we formulate the MUX design problem as a set problem, solve the problem with Sperner's theorem, and introduce the combinatorial coding strategy. In section 4, we propose a method to design reliability-aware CoMUXes. In section 5, we compare CoMUXes with classic MUXes and demonstrate the advantages of our method.

2 BACKGROUND

2.1 Classic Microfluidic Control Multiplexer

Different from the flow circuits for bio-experiments, a MUX uses the flow channels to address the control channels. Specifically, by pressurizing flow channels, we can provide individual access from the general control port to any control channel in the MUX.

A classic MUX uses $2N$ independent flow channels to address up to 2^N control channels. Figure 2 shows the structure of a small classic MUX. The flow channels and control channels inside the MUX are drawn with blue and green lines, respectively; valves are drawn as green rectangles. In this structure, channels of the same type are parallel to each other, and control channels are placed perpendicular to flow channels. By default, an intersection between control and flow channels cannot form a valve. Whether or not a valve should be constructed at an intersection depends on the coding strategy of the MUX.

The multiplexing mechanism in a classic MUX can be summarized as follows:

- Every control channel in the MUX is assigned with an index from 0 to $2^N - 1$, denoted as its binary representation. For example, in Figure 2, the index of control channel 3 is denoted as 011.
- Every two flow channels in the MUX are combined as a pair. A valve is implemented at the intersection between each control channel and each pair of flow channels. Specifically, the i^{th} bit of the control channel index decides the position of the valve between the control channel and the i^{th} pair of flow channels. If the bit is 0, the valve is at the intersection with the first flow channel; otherwise, the valve is at the intersection with the second flow channel.
- There are two potential pressure levels of each flow channel, i.e., depressurized, denoted as O, or pressurized, denoted as X. When a flow channel is depressurized, all valves along the flow channel will be open; on the other hand, when a flow channel is pressurized, all valves along the flow channel will be closed. For example, in Figure 2, open valves are marked with light green rectangles and closed valves are marked with crossings and gray rectangles.
- Each pair of flow channels must have opposite pressure levels. In particular, the pressure levels of the two flow channels form a control bit: (OX) represents 0 and (XO) represents 1. For example, in Figure 2, $(OX)(XO)(XO)$ form a control sequence 011, which will close at least one valve in all control channels except for control channel 3. In this manner, one can address control channel 3 via the general control port without affecting other control channels.

In general, a classic MUX uses a pair of flow channels to encode one bit of the index information, i.e., (OX) and (XO), so that $2N$ flow channels can encode 2^N indices of control channels. However, two flow channels can actually provide four combinations of the pressure levels, i.e., (OO), (OX), (XO) and (XX). If we can develop a coding strategy that takes advantage of the two wasted combinations, we will be able to address more control channels with the same resources.

2.2 Defects on Microfluidic Chips

Valves are fundamental components on microfluidic chips, and their functional correctness is essential for the chip. A valve consists of a flow channel segment, a control channel segment and a membrane

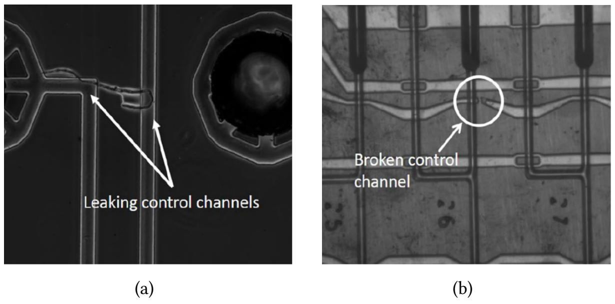


Figure 3: Common channel defects [16]. (a) Leakage. (b) Blockage.

in between. The flow channel segment of a valve has a round profile so that when the control channel segment of the valve is pressurized, it will push the membrane to fully block the flow channel. A valve malfunctions when there is a flaw in the valve itself, or when the control channel connected to the valve is defective.

On microfluidic chips, control channels usually have a smaller feature size than flow channels and are thus more prone to defects. In particular, there are two common types of control channel defects: leakage and blockage [15, 16].

- Leakage means that two control channels are accidentally connected, as shown in Figure 3(a). In this case, pressure may leak from one channel to the other, causing unexpected closure of valves.
- Blockage means that a control channel is accidentally broken, as shown in Figure 3(b). In this case, pressure can no longer pass the channel, causing valves along the channel not able to be closed.

In general, the longer a channel is, the more likely that a channel defect may happen. Besides, the probability of leakage between two neighboring channels increases, as the distance between the two channels decreases [15].

Since a MUX consists of many valves and control channels, reliability is an important metric to evaluate a MUX design. To improve the reliability, it is preferable to reduce the number of valves in the MUX, to shorten the control channels, and to enlarge the spacing distance between the control channels.

2.3 Problem Formulation

The problem can be formulated as follows:

Input :	*The number of to-be-addressed control channels*
Output :	*The design of a MUX*
Objective :	*1. Fully exploit the coding capacity*
	2. Reduce the number of valves
	3. Reduce the lengths and density of control channels

3 COMBINATORIAL CODING STRATEGY

3.1 Design Rules of the Coding Strategy

We define a *code* as a sequence of binary values $\{O, X\}$ assigned to a control channel in a MUX. The coding strategy of a MUX indicates the way that the MUX is constructed. Specifically, for a given control channel in a MUX:

- if the i^{th} bit of its code is O, there is a valve at the intersection between the control channel and the i^{th} flow channel;
- if the i^{th} bit of its code is X, there is no valve at the intersection between the control channel and the i^{th} flow channel.

For example, a classic MUX uses OX (0) and XO (1) to encode one bit of the control channel index, as introduced in Section 2.1. Thus, the code of control channel 3 is $OXXOXO$ (011), which indicates that this control channel has three valves at its intersections with the 1st, the 4th and the 6th flow channel, respectively, as shown in Figure 2.

When designing the coding strategy, there are two rules that we need to follow:

(1) Every control channel needs to have a unique code;
(2) When we trigger the code of a control channel, all valves along that control channel must be open, while at least one valve along every other control channel must be closed.

Here by *triggering a code*, we refer to the operation that we pressurize/depressurize the flow channels in the MUX according to the code, i.e., flow channels marked by O are depressurized, and flow channels marked by X are pressurized. For example, to trigger the code $OXXOXO$, we pressurize the 2nd, the 3rd and the 5th flow channels and depressurize the others. When we trigger a code, it naturally opens all valves along the control channel addressed by the code. Thus, in order to satisfy the coding rule (2), we just need to make sure that the code also closes at least one valve along every other control channel that is not addressed by that code.

To figure out the maximum number of control channels that N_F flows channels can address, we need to find out the maximum number of feasible codes of length N_F. To this end, for each code c, we construct:

- a set $\mathcal{T}_c \subseteq [1, N_F]$ containing the indices of the flow channels that are marked by X;
- a set $\overline{\mathcal{T}_c} = [1, N_F] \backslash \mathcal{T}_c$ containing the indices of flow channels that are marked by O.

For example, $\mathcal{T}_{OXXOXO} = \{2, 3, 5\}$; and $\overline{\mathcal{T}_{OXXOXO}} = \{1, 4, 6\}$. We can use c or $\mathcal{T}_c$ to represent a code interchangeably. In other words, when we pressurize all channels in $\mathcal{T}_c$, code c will be triggered. On the other hand, if any channel in $\overline{\mathcal{T}_c}$ is pressurized, at least one valve in the control channel addressed by c will be closed. In this manner, given $\mathcal{T}_c \neq \mathcal{T}_{c'}$, we can formulate the coding rule (2) as follows:

$$\forall \mathcal{T}_c \subseteq [1, N_F] : \ \forall \mathcal{T}_{c'} \subseteq [1, N_F] \ \exists x \in \mathcal{T}_c \bigcap \overline{\mathcal{T}_{c'}}. \tag{1}$$

Specifically, $\mathcal{T}_c$ can feasibly address a control channel, if for every other control channel (c') in the MUX, triggering c will pressurize a flow channel x. This leads to at least one valve in that control channel (c') being closed. Looking from the perspective of the sets, this constraint can be further formulated as:

$$\forall \mathcal{T}_c, \mathcal{T}_{c'} \subseteq [1, N_F] : \ \mathcal{T}_c \not\subseteq \mathcal{T}_{c'}. \tag{2}$$

In other words, for any two control channels in the MUX, their codes $\mathcal{T}_c$ and $\mathcal{T}_{c'}$ must not be subsets of each other.

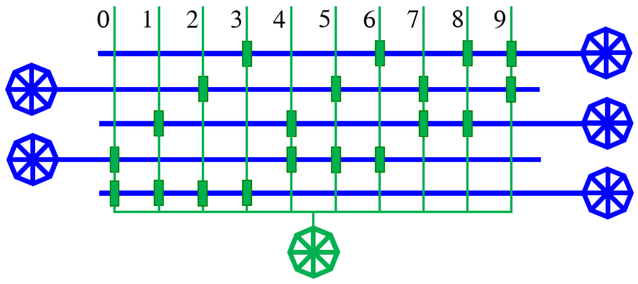


Figure 4: A MUX applying the combinatorial coding strategy can individually address 10 control channels with only 5 flow channels.

3.2 Sperner's Theorem and Combinatorial Coding Strategy

With the set model, we can see that the maximum coding capacity of N_F flow channels corresponds to the maximum number of subsets of $[1, N_F]$ that do not include one another. More precisely, given N_F flow channels, our target is to construct the largest possible set $\mathcal{S}$, with

$$S := \{\mathcal{T}_c \subseteq [1, N_F] \mid \forall \mathcal{T}_{c'} \in S : \mathcal{T}_c \nsubseteq \mathcal{T}_{c'}\}.$$

This problem is a famous set problem that can be solved with Sperner's Theorem [17]. In Sperner's Theorem, a family of subsets that do not belong to each other is called a *Sperner Family*. In other words, our target is to find the largest possible Sperner Family. According to Sperner's Theorem, given N_F elements in the set:

- For every Sperner family S, $|S| \leq C^{N_F}_{\lceil \frac{N_F}{2} \rceil} = C^{N_F}_{\lfloor \frac{N_F}{2} \rfloor}$.
- Equality in the above formula holds if and only if S consists of all subsets whose size is $\left\lceil \frac{N_F}{2} \right\rceil$, or all subsets whose size is $\left\lfloor \frac{N_F}{2} \right\rfloor$.

Thus, for a MUX with N_F flow channels, the maximum coding capacity is $C^{N_F}_{\lceil \frac{N_F}{2} \rceil}$, equivalent to $C^{N_F}_{\lfloor \frac{N_F}{2} \rfloor}$. This can be achieved by enumerating all $\mathcal{T}_c$ with $|\mathcal{T}_c| = \left\lceil \frac{N_F}{2} \right\rceil$ or all $\mathcal{T}_c$ with $|\mathcal{T}_c| = \left\lfloor \frac{N_F}{2} \right\rfloor$. It is noteworthy that the larger $|\mathcal{T}_c|$ is, the fewer valves will be needed on the channel. So we decide to enumerate all $\mathcal{T}_c$ with $|\mathcal{T}_c| = \left\lceil \frac{N_F}{2} \right\rceil$, which is more valve-saving. Since this coding strategy enumerates $\left\lceil \frac{N_F}{2} \right\rceil$-combinations out of N_F elements, we call it *combinatorial coding strategy.*

For example, the maximum coding capacity of $N_F = 5$ is $C^5_3 = 10$, i.e., 5 flow channels will be already enough to individually address 10 control channels in a MUX. To achieve the codes for each of the 10 control channels, we just need to enumerate all subsets of $\{1, 2, \cdots, 5\}$ with size 3, i.e., $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\cdots$, $\{3, 4, 5\}$. Based on the codes, we can construct a MUX, as shown in Figure 4. Specifically, $\{1, 2, 3\}$ corresponds to code $XXXOO$ and implies the leftmost control channel, which can be individually addressed by pressurizing the 1^{st}, the 2^{nd} and the 3^{rd} flow channels.

As we can see from the above example, different from the classic coding strategy which pairs every two flow channels together and sets them to either OX or XO, the combinatorial coding strategy allows neighboring flow channels to be set to both OO or XX to fully exploit the coding capacity. Besides, while a classic MUX requires an even number of flow channels, the combinatorial coding strategy also supports MUXes with an odd number of flow channels. This can be especially beneficial, when we need to increase N_f to address a little more control channels. For example, to address $2^3 + 1$ control channels, a classic MUX shown in Figure 2 has to increase the number of flow channels from 6 to 8; but to address $C^5_3 + 1$ control channels, the design shown in Figure 4 only needs to increase the number of flow channels from 5 to 6.

We compare the combinatorial coding strategy with the classic coding strategy for MUX designs at different scales, as shown in Figure 5. As we can see, the advantage of the combinatorial coding strategy becomes very impressive, as the design scale increases. For example, a classic MUX consisting of 20 flow channels can only address 1024 control channels, while a MUX applying the combinatorial coding strategy consisting of 20 flow channels can address 184, 756 control channels, which improves the coding capability by 180 times.

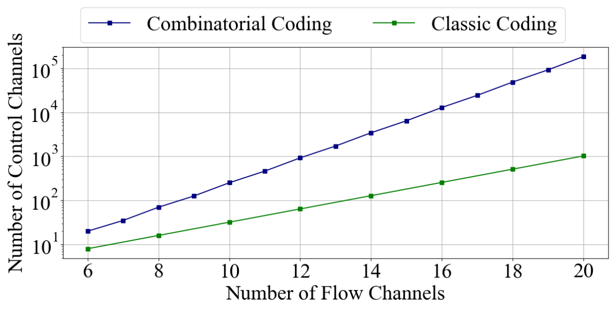


Figure 5: Comparison between different coding strategies. The x-axis represents the number of flow channels. The y-axis represents the maximum number of control channels that can be individually addressed by the MUX.

3.3 Improved Combinatorial Coding Strategy Considering Valve Reduction

With the combinatorial coding strategy, we require N_F flow channels to construct a MUX, when the number of the to-be-addressed control channels N_C falls in the range:

$$C^{N_F-1}_{\lceil \frac{N_F-1}{2} \rceil} < N_C \leq C^{N_F}_{\lceil \frac{N_F}{2} \rceil}.$$

When N_F is relatively large, the gap between $C^{N_F-1}_{\lceil \frac{N_F-1}{2} \rceil}$ and $C^{N_F}_{\lceil \frac{N_F}{2} \rceil}$ can be quite significant. For example, $C^9_5 = 126$ and $C^{10}_5 = 252$, which means that when the number of the to-be-addressed control channels is in the range from 127 to 252, 10 flow channels are needed to construct the MUX.

We notice that for N_C in a given range, although the number of flow channels is fixed, we may not need to implement the same number of valves to address the control channels. For example, suppose that $N_F = 10$. If $C^{10}_6 < N_C \leq C^{10}_5$, i.e., $210 < N_C \leq 252$, we need 5 valves along each control channel in the MUX. But when $C^9_5 < N_C \leq C^{10}_6$, i.e., $126 < N_C \leq 210$, we can construct up to 210 different sets of $\mathcal{T}_c$ of size 6 to address N_C control channels. As the size of $\mathcal{T}_c$ represents the number of flow channels marked by X, i.e., flow channels that do not have valves at their intersections with the corresponding control channel, the larger the size of $\mathcal{T}_c$ is, the fewer valves we need to implement along each control channel in the MUX. In other words, for $126 < N_C \leq 210$, we actually only need to implement 4 valves instead of 5 along each control channel, and thereby can reduce the total valve usage by 20%.

In general, given $C^{N_F-1}_{\lceil \frac{N_F-1}{2} \rceil} < N_C \leq C^{N_F}_{\lceil \frac{N_F}{2} \rceil}$, to address N_C control channels, we do not always need $\lceil \frac{N_F}{2} \rceil$ valves along each control channel. Instead, we only need k valves as long as $C^{N_F}_{N_F-k} \geq N_C$. Note that $C^{N_F}_{N_F-k} = C^{N_F}_{k}$.

4 RELIABILITY-AWARE COMUX DESIGN

To be distinguished from the classic MUX, we refer to a MUX applying the combinatorial coding strategy as a *CoMUX*. The combinatorial coding strategy allows one to enumerate a set of codes to individually address every control channel in a CoMUX. However, the strategy does not specify the order of the control channels. Thus, based on the combinatorial coding strategy, we can design different CoMUXes by exchanging the order of the control channels. In this section, we propose a method to design the CoMUX. so that the usage of valves and control channels in the CoMUX is minimized, and thus the reliability of the CoMUX can be improved.

4.1 Selection of N_F and k

Given a number of to-be-addressed control channels, denoted as N_C, we first need to determine the number of flow channels in the CoMUX, denoted as N_F, and the number of valves along each control channel of the CoMUX, denoted as k. Specifically, we choose the smallest possible N_F with $C^{N_F}_{\lceil \frac{N_F}{2} \rceil} \geq N_C$, and then the smallest possible k with $C^{N_F}_{k} \geq N_C$.

4.2 Potential of channel merging

Next, we construct an initial CoMUX design. The initial CoMUX has a crossbar structure, which is similar to a classic MUX. Figure 4 shows an example CoMUX with $N_F = 5$ and $k = 2$. We notice that, some part of adjacent control channels can partially merge to save valves and control channel lengths. Specifically, considering $1 \leq i \leq N_F$, two control channels can partially merge from the intersections with the i^{th} flow channel to the bottom of the MUX, as long as the code bits from the i^{th} bit to the N_F^{th} bit are pairwise identical. Take the 7^{th} and the 8^{th} control channels in Figure 4 as an example. The code of the 7^{th} control channel is $XOOXX$, and the code of the 8^{th} control channel is $OXOXX$. Since their last three code bits (OXX) are identical, we can combine the channel segments corresponding to these three bits to save one valve and to save the channel length.

4.3 Algorithm for code generation

We develop a greedy algorithm to determine the codes of the control channels from left to right in the initial CoMUX. The key idea of the algorithm is to make the identical bottom parts of two control channels as large as possible.

Instead of directly working on the codes of the control channels, our algorithm works on the inverse of the codes. We define the inverse code of code c as:

$$I_c := \{N_F - a + 1 \mid \forall a \in \overline{\mathcal{T}_c}\}.$$

which indicates the location of the valves along the channel, but with a reverse index order.

In our algorithm, we first build up a list MEM of size k to enumerate the inverse codes. Suppose $val(i)$ indicates the value stored in the i^{th} element. MEM is initialized with [1,2,...,k], i.e., $val(i) = i$, as our first inverse code. Then, we produce the next inverse code based on the current inverse code stored in MEM, iteratively.

There are two functions in our algorithm: $INCREASE$ and $RESET$. For $INCREASE$, our algorithm always attempts to increase $val(k)$ by one. For $RESET$, our algorithm examines $val(i)$ sequentially in a reverse order from $i = k$ to $i = 1$. If there is any specific j^{th} element such that $val(j) + 1 < val(j + 1)$, we set all the $val(i)$ from $i = j$ to $i = k$ to $val(j) + 1$, $val(j) + 2$, ..., $val(j) + k - j + 1$.

Our algorithm performs $INCREASE$ and $RESET$ iteratively. For each iteration, We repeatedly perform $INCREASE$ until $val(k)$ is equal to N_F, and then $RESET$ follows. The entire algorithm terminates when that last $RESET$ is unable to find any specific j^{th} element such that $val(j) + 1 \neq val(j + 1)$. For each time $INCREASE$ or $RESET$ is successfully executed, we obtain a valid inverse code, which can be transformed back into a standard code.

4.4 Recursion relation of combination function

We propose to employ the recursion relation of the combination function to efficiently generate the codes of the initial CoMUX. The recursion relation of $C^{N_F}_{k}$ is:

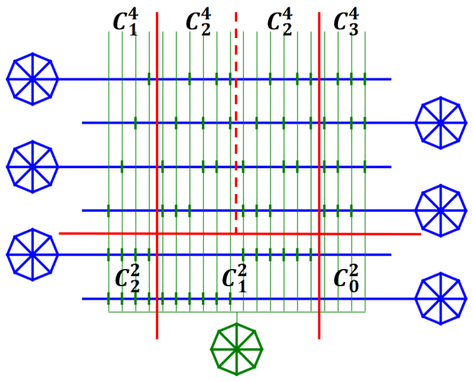
$$\begin{aligned} C^{N_F}_{k} &= C^{N_F-1}_{k} + C^{N_F-1}_{k-1} \\ &= C^{N_F-2}_{k} + 2C^{N_F-2}_{k-1} + C^{N_F-2}_{k-2} \\ &= C^{N_F-3}_{k} + 4C^{N_F-3}_{k-1} + 4C^{N_F-3}_{k-2} + C^{N_F-3}_{k-3} \\ &= \sum_{i=0}^{k-1} C^{k-1}_{i} \cdot C^{N_F-(k-1)}_{k-i} \end{aligned} \tag{3}$$

where C^{k-1}_{i} is the binomial coefficient. It is worth mentioning that $N_F - (k-1) \geq k$ is tautology, considering that the largest possible value of k is $\left\lceil \frac{N_F}{2} \right\rceil$.

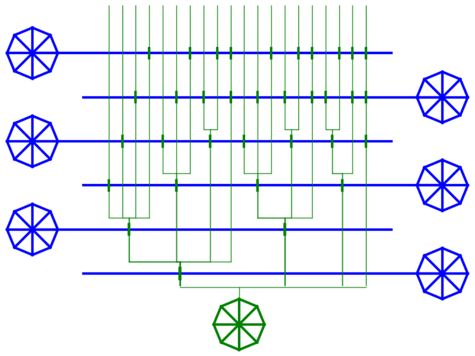
With (3), we can break $C^{N_F}_{k}$ into the summation of a series of terms, and each term is a multiplication of two combination functions. To generate the codes, we just need to apply the algorithm that we propose in Section 4.3 to derive the inverse code for each combination function.

Consider an example with $N_C = 20$, $N_F = 6$ and $k = 3$. $C^{N_F}_{k} = C^{6}_{3} = C^{2}_{0} \cdot C^{4}_{3} + C^{2}_{1} \cdot C^{4}_{2} + C^{2}_{2} \cdot C^{4}_{1}$. With the introduced algorithm, we can build an initial MUX design as shown in Figure 6(a).

For each multiplication of two combination functions, i.e., $C^{k-1}_{i} \cdot C^{N_F-(k-1)}_{k-i}$, the first function (binomial coefficient) determines the first $k-1$ bits of the inverse code, and the second function determines the last $N_F - (k-1)$ bits of the inverse code. When the code



(a)



(b)

Figure 6: CoMUX design of $N_C = 20$. (a) The Initial CoMUX. (c) Optimized CoMUX.

numbers of the two functions are both greater than one, we enumerate the second term first, such as C_2^4 in $C_1^2 \cdot C_2^4$ in the example shown in Figure 6(a), for the ease of channel merging.

4.5 Channel merging

Though our algorithm for code generation is inherently friendly for channel merging, we still need to explicitly determine which group of channels and how long the channel segments should merge.

We construct a list of size N_C to record the group information of each control channel. Initially, all control channels are set to belonging to the same channel group.

We start the merging from the bottom and move row by row, i.e., per flow channel, to the top. Two valves can merge together if (1) they are along the same flow channel, (2) they are on the adjacent control channels, and (3) they still belong to the same channel group. For a specific flow channel, if a control channel forms a valve with it, while another control channel forms no valve with it, these two channels will be assigned to different channel groups. It is worth mentioning that two adjacent valves cannot merge if they already belong to different groups.

Figure 6(b) shows the optimized CoMUX after channel merging. Comparing with the initial CoMUX shown in Figure 6(a), we save 26 valves, i.e., 43.3% of the original valve usage, and 31.7% total length of control channels.

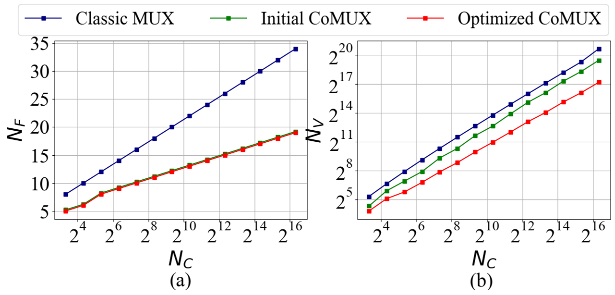


Figure 7: Variation of performance metrics with N_C for the classic MUX and CoMUX. (a) N_F. (b) The number of valves.

5 EXPERIMENTAL RESULTS

To demonstrate the performance of our method, we compare CoMUX to the classic MUX with two groups of experiments.

5.1 Experiments on resource usage

Given a number of to-be-addressed control channels, denoted as N_C, we evaluate the resource usage of CoMUX and the classic MUX based on two metrics: the number of flow channels, denoted as N_F, and the number of valves, denoted as N_V. In particular, the number of flow channels is equal to the number of flow ports, which also represents the number of the required external pneumatic controllers. Thus, more flow channels not only indicates more area consumption, but also a larger chip-to-world interface.

Our experiment starts from $N_C = 10 \times 2^0$, and we double N_C for every comparison until N_C reaches 10×2^{13}, which is enough to satisfy the control demand of current microfluidic applications. The results of the comparisons are shown in Table 1 and Figure 7.

In general, CoMUX enables 37.5%–44.1% reduction in the number of flow channels. That is because a CoMUX can address $C^N_{\lceil \frac{N}{2} \rceil}$ control channels with N flow channels, while with the same number of flow channels, a classic MUX addresses at most 2^N control channels. Since $C^N_{\lceil \frac{N}{2} \rceil}$ grows much faster than 2^N, as shown in Figure 5, the reduction in flow channel usage becomes more significant, as the number of to-be-addressed control channels increases.

As for valve usage, since the combinatorial coding strategy contributes to a remarkable reduction in the number of flow channels, and each flow channel incorporates many valves, an initial CoMUX design without structural optimization can already achieve around 40%–60% reduction in the number of valves. With our design method proposed in section 4, the optimized CoMUX can further reduce the valve usage by up to 80% compared to the initial CoMUX design. The reduction becomes more significant, as the number of to-be-addressed control channels increases. In particular, for the largest case, the optimized CoMUX only requires 10.8% of the valves compared to a classic MUX.

5.2 Experiments on reliability against manufacturing defects

We evaluate the reliability of CoMUX and the classic MUX based on three metrics: the probability of valve defects, the probability of control channel blockage defects, and the probability of control channel leakage defects. Same as the settings in section 5.1, we

Table 1: Resource Usage in different MUXes under different control demands.

Mux Type	Performance Metrics		Number of Control Channels (N_C)													
			10	20	40	80	160	320	640	1280	2560	5120	10240	20480	40960	81920
Classic MUX	Number of Flow Channels (N_F)		8	10	12	14	16	18	20	22	24	26	28	30	32	34
	Number of Microvalves (N_V)		40	100	240	560	1280	2880	6400	14080	30720	66560	143360	307200	655360	1392640
CoMUX	Number of Flow Channels (N_F)		5	6	8	9	10	11	12	13	14	15	16	17	18	19
	Number of Microvalves (N_V)	initial	20	60	120	240	640	1280	3200	6400	15360	35840	71680	163840	321680	737280
		optimized	14	34	55	110	234	459	982	1986	4142	8806	17132	36392	71011	150489

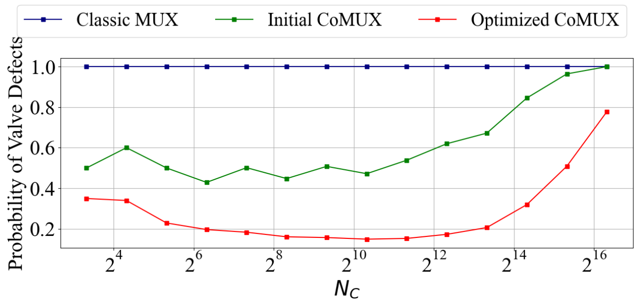


Figure 8: The (relative) probability of different MUXes to suffer from valve defects.

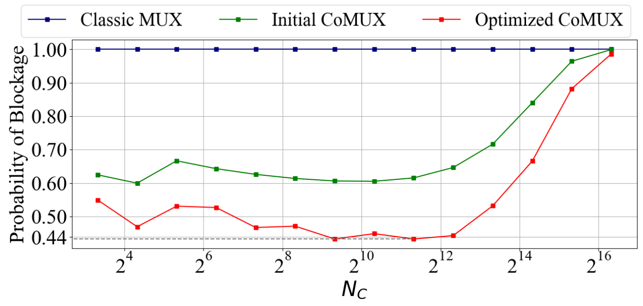


Figure 9: The (relative) probability of different MUXes to suffer from blockage defects.

start from $N_C = 10 \times 2^0$, and double N_C for every comparison until N_C reaches 10×2^{13}. Since the risk of the blockage and leakage defects is related to the lengths and the spacing distance of the channels, we assume some physical parameters for the designs of the MUXes. Specifically, we set the widths of the flow and the control channels to $100\mu m$ and $30\mu m$, respectively. Besides, we set the distance between two flow channels as $1300\mu m$, considering the area consumed by the flow ports; and we set the distance between two control channels as $400\mu m$.

5.2.1 *Performance against valve defects*

We assume that valve defects are independent of channel defects, and denote the probability that a defect happens to a random valve as p_{valve}. Then, we calculate the probability P_v that a MUX suffers at least one defective valve as follows:

$$P_v = 1 - (1 - p_{valve})^{N_V}. \tag{4}$$

In our experiments, p_{valve} is set to 1×10^{-5}.

We consider the classic MUX as the baseline design and denote the probability that a classic MUX suffers at least one defective valve as 1, Thus, the relative probability of a CoMUX that suffers at least one defective valve can be calculated as the ratio of P_v of the CoMUX to P_v of the classic MUX.

The results of the experiments are shown in Figure 8. When $N_C < 4000$, the probability of an initial CoMUX to suffer from valve defects is always less than 60% of the baseline, and the probability of an optimized CoMUX to suffer from valve defects is even much lower than the intitial CoMUX, thanks to the further reduced number of valves. As N_C becomes larger, the number of valves increases quickly and thus all MUXes are more prone to defects. Nevertheless, the optimized CoMUX is still more reliable than the other MUXes.

5.2.2 *Performance against blockage defects*

As introduced in Section 2.2, the risk of blockage defects is positively correlated to the channel length. To test the performance of different MUXes against blockage defects, we divide the control channel segments of the MUXes into small grid cells of size $30\mu m \times 30\mu m$. For each grid cell, we model its probability of suffering a blockage defect. Specifically, we consider a grid cell to suffer a blockage defect, if any cell along the control channel between this cell and the control port suffers a blockage defect. In other words, the farther away a grid cell is from the control port, the more likely the grid cell is to suffer a blockage defect.

We denote the probability of a blockage randomly happening to a cell as $p_{blockage}$. In our experiment, $p_{blockage}$ is set to 1×10^{-7}. With this setting, the probabilities for classic MUXes with 10, 20 and 30 flow channels to have blockage defects are 0.088%, 5.5% and 93.3%, respectively. We consider the classic MUX as the baseline design and denote the probability that a classic MUX suffers at least one blockage defect as 1, and we calculate the relative probability of the initial as well as the optimized CoMUXes to suffer blockage defects corresponding to the baseline.

Figure 9 shows the results of the comparison. When $N_C < 4000$, thanks to the reduced channel lengths, the initial CoMUX reduces the risk of blockage by around 40% compared to a classic MUX. Moreover, the optimized CoMUX reduces the risk by up to 56% compared to a classic MUX, thanks to the merging of control channels in the bottom part of the CoMUX. As N_C becomes very large, i.e., more than 20000, although the optimized CoMUX still has a better performance, all MUXes become quite prone to defects. The high defective rate results from the large number of control channels, which cannot be simply addressed by optimizing the structure of the MUXes. This also indicates that reliability may be a critical issue for the scale up of microfluidic chips.

To demonstrate the distribution of the risk of blockage defects in different MUXes, we synthesize three MUXes of different types with $N_C = 20$, and show the risk distribution as a heapmap in Figure 10(a). Specifically, the control port is at the bottom of the MUXes but not displayed in Figure 10(a). We can see that the risk of blockage increases, as the distance between the channel segment and the control port increases.

5.2.3 *Performance against leakage defects*

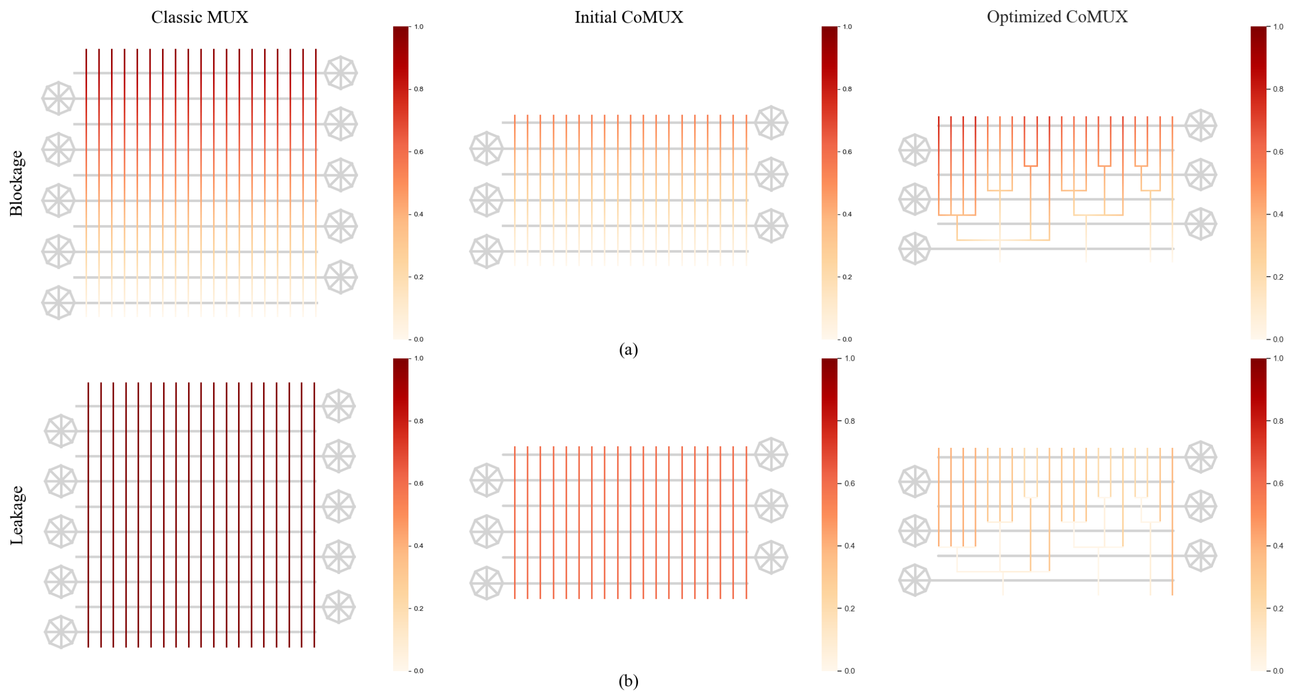


Figure 10: Heatmap of probabilities of defects in different types of MUX when $N_C = 20$.

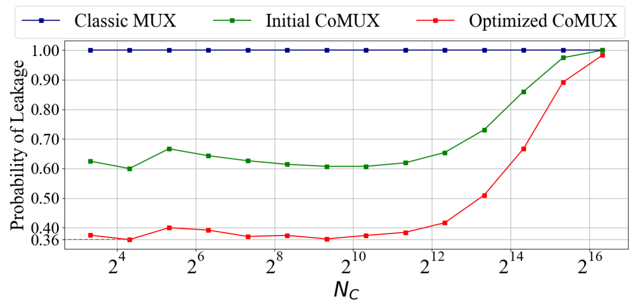


Figure 11: The (relative) probability of different MUXes to suffer from leakage defects.

As introduced in section 2.2, the probability of leakage between two control channels increases, as the distance between the two control channels decreases. Besides, as the channel length increases, the risk of leakage increases. Nevertheless, the distance between the two channels dominates the risk of leakage.

To test the performance of different MUXes against leakage, we take each straight control channel segment in the MUX as the unit of calculation. We assume that a control channel will only have leakage with its neighboring control channel, and we model the probability of leakage between two channels as:

$$P_l = p_{leakage} \times channel\ length/distance^2 \tag{5}$$

$p_{leakage}$ is the unit probability, which is set to 1×10^{-6} in our experiment. With this setting, the probabilities for classic MUXes with 10, 20 and 30 flow channels to have leakage defects are 0.097%, 6.1% and 95.0% respectively. Still, we consider the classic MUX as the baseline design and denote the probability that a classic MUX suffers at least one leakage defect as 1, and we calculate the probability of the initial as well as the optimized CoMUXes to suffer leakage defects corresponding to the baseline.

Figure 11 shows the results of the comparison. In general, similar to the experiments regarding blockage, both CoMUXes perform better than the classic MUX, and the optimized CoMUX is more reliable than the initial MUX. Still, as N_C becomes very large, all MUXes will be prone to defects. Considering that current microfluidic applications can be satisfied with a few hundreds of control channels, CoMUXes show significant advantages in reliability compared to classic MUXes.

Figure 10(b) demonstrates the distribution of the risk of leakage defects in different MUXes with $N_C = 20$. As we can see, the risk of leakage defects distribute averagely in the classic MUX and in the initial CoMUX. Since the optimized CoMUX significantly reduces the density of the control channels close to the bottom control port, the risk of leakage in the bottom part of CoMUX is quite small.

6 CONCLUSION

In this paper, we have proposed a new multiplexer, named CoMUX, to address the valve control problem in flow-based microfluidic chips. CoMUX employs combinatorial coding strategy based on Sperner's theorem to achieve the maximum coding capacity. With N_F flow channels, CoMUX can achieve the proven maximum coding capacity, i.e., $C^{N_F}_{\lceil \frac{N_F}{2} \rceil}$, which is much larger than $2^{\frac{N_F}{2}}$ supported by a classic MUX. We have also proposed a method for automatic CoMUX synthesis. To improve the efficiency of the synthesis, we apply the recursion relation of combination functions. Our method enables the merging of control channels and valves and thus improves the reliability of the design. Experimental results have confirmed that CoMUX significantly reduces the chip-to-world interface and outperforms the classic MUX in terms of resource usage and reliability.

REFERENCES

[1] K. Hu, K. Chakrabarty, and T.-Y. Ho, "Computer-Aided Design of Microfluidic Very Large Scale Integration (mVLSI) Biochips.", Springer, 2017.
[2] J. M. Perkel, "Life Science Technologies: Microfluidics-Bringing New Things to Life Science.", *Science*, 322.5903, (2008): 975–977.
[3] D. Erickson, D. Li, and U. J. Krull, "Modeling of DNA Hybridization Kinetics for Spatially Resolved Biochips.", *Analytical biochemistry*, 317.2, (2003): 186–200.
[4] H. Yao, Q. Wang, Y. Ru, T.-Y. Ho, and Y. Cai, "Integrated Flow-Control Co-Design Methodology for Flow-Based Microfluidic Biochips," IEEE Design and Test of Computers (IEEE DT), vol. 32, no. 6, pp. 60-68, December 2015.
[5] M. Kock, A. Evans, A. Brunnschweiler, "Microfluidic Technology and Applications.", *Research Studies Press*, Hertfordshire, UK, 2000.
[6] X. Huang, T.-Y. Ho, W. Guo, B. Li, U. Schlichtmann, "MiniControl: Synthesis of Continuous-Flow Microfluidics with Strictly Constrained Control Ports.", *The 56th Annual Design Automation Conference (DAC)*, 2019.
[7] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Storage and Caching: Synthesis of Flow-based Microfluidic Biochips," IEEE Design and Test of Computers (IEEE DT), vol. 32, no. 6, pp. 69-75, December 2015.
[8] T.-M. Tseng, B. Li, M. Li, T.-Y. Ho, U. Schlichtmann, "Reliability-Aware Synthesis with Dynamic Device Mapping and Fluid Routing for Flow-Based Microfluidic Biochips.", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35.12: 1981–1994.
[9] M. Shayan, S. Bhattacharjee, Y.-A. Song, K. Chakrabarty, R. Karri, "Toward Secure Microfluidic Fully Programmable Valve Array Biochips.", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27.12: 2755–2766.
[10] T. Thorsen, S. J. Sebastian, and S. R. Quake, "Microfluidic large-Scale Integration.", *Science*, 298.5593, (2002): 580–584.
[11] L. M. Fidalgo and S. J. Maerkl, "A Software-Programmable Microfluidic Device for Automated Biology.", *Lab on a chip*, 2011, 11.
[12] Q. Wang, S. Zuo, H. Yao, T.-Y. Ho, B. Li, U. Schlichtmann, Y. Cai, "Hamming-Distance-Based Valve-Switching Optimization for Control-Layer Multiplexing in Flow-Based Microfluidic Biochips.", *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 524–529.
[13] Y. Zhu, X. Huang, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, U. 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.
[14] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, U. Schlichtmann, "Columba S: A Scalable Co-Layout Design Automation Tool for Microfluidic Large-Scale Integration.", *The 55th Annual Design Automation Conference (DAC)*, 2018.
[15] K. Hu, F. Yu, T.-Y. Ho, K. 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.
[16] K. Hu, T.-Y. Ho, K. Chakrabarty, "Test Generation and Design-for-Testability for Flow-Based mVLSI Microfluidic Biochips." *2014 IEEE 32nd VLSI Test Symposium (VTS)* IEEE, 2014.
[17] E. Sperner, "Ein Satz über Untermengen einer endlichen Menge.", *Mathematische Zeitschrift*, 27 (1928): 544–548.