I, Xinyu Sun, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Engineering.

It is entitled:
Fault Modeling and Fault Type Distinguishing Test Methods for Digital Microfluidics Chips

Student's name: Xinyu Sun

This work and its defense approved by:

Committee chair: Wen Ben Jone, Ph.D.

Committee member: Xingguo Xiong, PhD

Committee member: Ian Papautsky, Ph.D.
Fault Modeling and Fault Type Distinguishing Test Methods for Digital Microfluidics Chips

A Thesis submitted to the Graduate School of the University of Cincinnati in partial fulfillment of the requirements for the degree of Master of Science in School of Electronics and Computing Systems of College of Engineering and Applied Science

Aug. 14, 2013

Xinyu Sun

M.S, EE, University of Bridgeport

Committee Chair: Dr. Wenben Jone, Ph.D
Abstract

Physical defects in digital microfluidics chips (DMCs) can be very complicated and extremely difficult to find precise models, because each defect may occur anywhere. In this thesis, we develop high-level abstract fault models based on investigating the faulty and fault-free behaviors of droplet moving. Two new fault models that were not found previously are proposed to enhance the reliability of DMCs. We believe that the high-level fault models can completely cover all defects involving two cells in a DMC array. Based on the new high-level fault models, we propose march algorithms (march-d and march-p/p+) to generate test patterns that can detect and distinguish fault types for each faulty digital microfluidics chip. This is accomplished by merging both march-d and part of march-p without causing too much test length increase. These algorithms are implemented into a FPGA board attached to the simulated digital microfluidics chip such that built-in self-test can be accomplished without human intervention. We also develop an EDA tool and simulation platform for the proposed DMC-BIST system. Experimental results demonstrate that the proposed fault models, test and fault type distinguishing methods, built-in self-test circuit design, and emulation tool can effectively and efficiently achieve high quality test with minimal test cost.
Acknowledgments

Firstly, I would like to highly thank my advisor, Dr. Wenben Jone for his great help and guidance throughout my thesis. I would also like to thank you for offering the Low power VLSI and VLSI Testing classes so useful and important.

I would also like to greatly thank to Dr. Xinguo Xiong, who gives me consistent encourage and good ideas during my thesis work and for being in my thesis committee in spite of his busy schedule.

I would like to thank to Dr. Ian Papautsky for offering me great help during my thesis’s experiments without any reservations and spending his valuable time in being my thesis committee and reviewing this work.

I would like to thank my parents for always supporting me throughout my life. I don’t really know how much to thank you for all the sacrifices you have made for me.

Sincere thanks to Yuguang Liu for her constant help during my thesis work and experiment. I would also like to extend my thanks to my girlfriend, my lab mates and all other friends. The days we spent studying hard for exams and working on various projects to our discussion on thesis are unforgettable.
Contents

Abstract ii

Acknowledgments iv

List of Figures viii

List of Tables xi

1 INTRODUCTION 1

1.1 Digital Microfluidics Chip Overview .............................................................. 1

1.2 Importance of Testing Techniques for DMC ...................................................... 2

1.3 Research Objective and Thesis Organization ................................................. 4

2 BACKGROUND 6

2.1 DMC Working Principle and Implementation .................................................. 6

2.2 DMC Operations ............................................................................................. 8

2.3 Related Prior Work ......................................................................................... 14

3 FAULT MODELING 18

3.1 FX Fault Modeling .......................................................................................... 19

3.1.1 Fundamental Concept ............................................................................... 19
3.1.2 Fault Analysis .............................................................. 20
3.1.3 Experimental Support ................................................... 22

3.2 PFX Fault Modeling .......................................................... 23
  3.2.1 Fundamental Concept ................................................ 23
  3.2.2 Fault Analysis ........................................................... 24
  3.2.3 Experimental Support ................................................. 25

3.3 PFXy Fault Modeling .......................................................... 26
  3.3.1 Fundamental Concept ................................................ 26
  3.3.2 Fault Analysis ........................................................... 27
  3.3.3 Experimental Support ................................................. 28

4 TEST STRATEGY ................................................................. 30
  4.1 March-p Test ................................................................. 32
    4.1.1 Test Pattern Generation ........................................... 32
    4.1.2 Test Analysis .......................................................... 34
  4.2 March-p+ Test ............................................................... 37
  4.3 March-d Test ................................................................. 38
    4.3.1 Test Pattern Generation ........................................... 38
    4.3.2 Test Analysis .......................................................... 45
  4.4 Fault Type Distinguishable Test Strategy ....................... 47
    4.4.1 Motivation .............................................................. 47
    4.4.2 Test Pattern Generation for Distinguishing Fault Types ....... 49
    4.4.3 Test Analysis .......................................................... 52
  4.5 Discussion ......................................................................... 53

5 BUILT-IN SELF-TEST DESIGN AND EMULATION .................. 58
  5.1 Why Built-in Self-test ..................................................... 58
5.2 Architecture Design of DMC with BIST ........................................... 59

5.3 Implementation and Emulation................................................................. 61

  5.3.1 Display Design ........................................................................ 62

  5.3.2 BIST Circuit Design ................................................................. 64

  5.3.3 Emulation ............................................................................. 70

6 CONCLUSION AND FUTURE WORK ........................................... 72

BIBLIOGRAPHY ........................................................................ 75
List of Figures

Fig. 1.1 Fabricated digital microfluidic arrays [1] ................................................................. 2
Fig. 1.2 Operations of digital microfluidic chips [2] ............................................................ 2
Fig. 2.1 Electrowetting-on-dielectric (EWD) effect [2] ....................................................... 6
Fig. 2.2 Working principle and structure of a DMC [2] .................................................... 7
Fig. 2.3 Top view of digital microfluidic array [23] ............................................................ 8
Fig. 2.4 Behavior of droplet transportation ........................................................................ 10
Fig. 2.5 Droplet dispensing phenomenon .......................................................................... 11
Fig. 2.6 Side view of split operation .................................................................................. 12
Fig. 2.7 Top view of split operation .................................................................................. 12
Fig. 2.8 Top view of merge operation .............................................................................. 13
Fig. 2.9 (a) Pivots and (b) routing path for a 2×4 microfluidic mixer ............................... 13
Fig. 3.1 FX fault model ........................................................................................................ 19
Fig. 3.2 Comb broken (missing) fault ................................................................................ 21
Fig. 3.3 FX fault model experiment [22] ......................................................................... 22
Fig. 3.4 PFX fault model .................................................................................................... 23
Fig. 3.5 PFX fault model experiment [22] .......................................................... 25
Fig. 3.6 Physical fault injection for PFX fault .................................................. 25
Fig. 3.7 PFEn fault model ............................................................................. 26
Fig. 3.8 PFXy fault location analysis ............................................................. 28
Fig. 3.9 PFWn fault model experiment [22] .................................................. 28
Fig. 3.10 Physical fault injection for PFWn fault ............................................. 29

Fig. 4.1 A test route .................................................................................... 31
Fig. 4.2 March-p algorithm description ....................................................... 32
Fig. 4.3 Illustration of march-p algorithm ....................................................... 33
Fig. 4.4 Cell (i,j) .................................................................................... 35
Fig. 4.5 Example of PFX fault ..................................................................... 35
Fig. 4.6 Example of PFXy fault .................................................................. 36
Fig. 4.7 March-p+ algorithm ........................................................................ 37
Fig. 4.8 Illustration of march-p+ ................................................................. 38
Fig. 4.9 Diagonal paths ............................................................................... 39
Fig. 4.10 Path covered by all diagonal paths ............................................... 39
Fig. 4.11 Outer cell routing ......................................................................... 40
Fig. 4.12 Special case ............................................................................... 41
Fig. 4.13 March-d algorithm ........................................................................ 42
Fig. 4.14 Illustration of march-d test .............................................................. 43
Fig. 4.15 Illustration of test time ................................................................. 43
Fig. 4.16 Application of march-d algorithm ................................................................. 45
Fig. 4.17 PFXy fault simulation for diagonal path ...................................................... 46
Fig. 4.18 Fault simulation experiment for PFWn under march-d [22] ............................. 47
Fig. 4.19 Example of misdiagnosis .............................................................................. 48
Fig. 4.20 Fault type distinguishing by different algorithms .......................................... 50
Fig. 4.21 Fault type distinguishing test generation ...................................................... 50
Fig. 4.22 One way detection for PFXy faults ............................................................... 51
Fig. 4.23 Illustration of flooding issue ........................................................................ 54
Fig. 4.24 Illustration of cell testability ......................................................................... 55
Fig. 4.25 DMC DFT structure design ......................................................................... 56

Fig. 5.1 General BIST techniques [29] ........................................................................ 59
Fig. 5.2 Block diagram of a BIST system .................................................................... 59
Fig. 5.3 Capacitive sensing circuit [15] ....................................................................... 61
Fig. 5.4 Emulation Platform ....................................................................................... 62
Fig. 5.5 BIST circuit block diagram ........................................................................... 65
Fig. 5.6 Schematic of BIST logic ............................................................................... 67
Fig. 5.7 (a) March-d algorithm simulation .................................................................... 68
Fig. 5.7 (b) March-p algorithm simulation .................................................................... 68
Fig. 5.8 RTL diagram of BIST blocks ......................................................................... 70
Fig. 5.9 Emulation result ......................................................................................... 70
List of Tables

Table 2.1 Physical defects summary [8-16] ................................................................. 14

Table 2.2 Classic test strategies ................................................................................. 15

Table 4.1 The march-p algorithm ............................................................................. 33

Table 4.2 Fault type identification analysis .............................................................. 52

Table 5.1 Cells array design ..................................................................................... 63

Table 5.2 Defective cell design .................................................................................. 63

Table 5.3 Droplet design ........................................................................................... 64

Table 5.4 Verilog code of generating reg_x and reg_y ............................................ 66

Table 5.5 Partial Verilog code of merging test patterns ........................................... 69
Chapter 1

Introduction

1.1 Digital Microfluidic Chip Overview

Digital microfluidics technology is one of the best technologies for lab-on-a-chip system design using the well-know electrowetting effect to conduct the movement of nanoliter discrete droplets on a two dimensional microfluidic array as shown in Fig. 1.1. Different kinds of biomedical operations such as dispense, mixing, split, merge (Fig. 1.2) can be achieved by digital microfluidic chips (DMCs) through controlling different voltages applied to their electrodes.

Microfluidics chips are composed of an emerging category of mixed technology microsystems [2]. These microsystems, referred to interchangeably in DMCs, lab-on-a-chip and bioMEMS [2-3], are much possible to clinical diagnostics, drug discovery and so on. Such applications can benefit from the small size of biochips, lower cost, and higher sensitivity compared to conventional laboratory methods [2].
1.2 Importance of Testing Techniques for DMCs

Research in IC testing has extended from digital circuit test to analog and mixed-signal devices test in the past several decades. More recently, new test techniques for
mixed-technology microelectromechanical systems (MEMS) are also getting attention [3-4]. Fabrication techniques such as silicon micromachining lead to new types of manufacturing defects in MEMS [5]. The level of system integration and the complexity of DMCs are expected to drastically increase in the near future because of the increasing need for multiple and concurrent bioassays on a chip [6]. However, many factors such as shrinking processes, new materials, will make these DMCs more susceptible to manufacturing defects. Critical to the deployment of DMCs in such diverse areas is the dependability (reliability) of these systems. Thus, robust on-line and off-line testing techniques are required to ensure the system dependability. Furthermore, reconfiguring a defective digital microfluidic chip to be functional is crucial to increase the fabrication yield (thus reducing the manufacturing cost) [2].

Physical defects in digital microfluidics chips can be very complicated and difficult to find their precise models, and each defect may lead to different fault behavior. Recently, most digital microfluidics chip test research papers [6-21] are based on physical defects (e.g., short circuit between two adjacent cells, open circuit of electrodes, cell contamination, etc.) As a matter of fact, if we want to fully explore a digital microfluidics chip’s fault models according to real physical defects, it is very hard and inadequate, since it is impossible to consider all complex physical defects. For example, when a pair of diagonal electrodes is completely or partially shorted, a coupling fault (which was not developed before) will occur between these two corresponding cells. Thus, the test strategies developed (e.g., euler path algorithm, scan-like parallel testing) based on those physical fault models cannot guarantee to detect all defects and deliver fault-free chips. Therefore, we develop high-level fault models that can
reflect the behavior of all different defects, and the models are computationally efficient in terms of fault simulation and test pattern generation.

1.3 Research Objectives and Thesis Organization

In this thesis, we develop high-level fault models based on investigating the faulty and fault-free behavior of droplet moving. Since no matter what a physical defect is, finally it can be mapped into a high-level model. If we consider all behaviors of a fault-free droplet, we definitely cover and detect all the corresponding physical defects. It is obvious that the droplet behavior is limited and deterministic, while the number of physical defects is almost unlimited. The high-level fault models we build in this thesis dramatically increase the diversity of fault cases that were not covered by physical defects defined in previous works [8-16]. Although, we cannot formally prove, we strongly believe that our proposed high-level fault models exhaustively covers all defects involving two cells in a DMC array.

Based on the new high-level fault models, we propose march algorithms (march-d and march-p/p+) to generate test patterns that can detect and distinguish fault types for a faulty digital microfludic chip without causing flooding problems. The beauty of this test and fault type distinguishing test method is to skillfully merge both march-d and part of march-p without increasing too much test time. These algorithms have been implemented into an FPGA attached to a simulated DMC such that built-in self-test can be accomplished without human intervention. We have also developed an EDA tool and simulation platform for a DMC-BIST
system.

Our high level fault modeling and test strategies might not directly applicable to industry devices condition and could mainly contribute to theoretically analysis especially for test pattern application in this research, because currently the devices could be easily destroyed during testing and thus cannot be used for normal operations. In addition, it is hard to control the droplet size in current device technology. However, we highly believe it will be used in the near future, when the mass productivity for DMC devices occurs. Therefore, the research for this area is necessary and can be used in the future.

Obviously, a good research cannot deal with theoretical analysis without any experimental support. There are three real-chip experiments in our research conducted by in [22] to support and verify the existence of fault models, and PFXy fault missed detection by the march-d algorithm. The rest of the thesis is organized as follows:

- In Chapter 2, we will briefly introduce the background of digital microfluidics working principles and present the related works in DMC fault modeling and testing.
- In Chapter 3, detailed high-level faults modeling will be discussed through real chip verification.
- Test strategies including test pattern generation, marching algorithm fault type distinguishing test method design, and fault coverage analysis will be covered in Chapter 4.
- In Chapter 5, we describe the BIST circuit design and DMC-BIST system emulation; a simple and yet efficient EDA tool is developed.
- Finally, in Chapter 6 we conclude our results and suggest several issues for further research.
Chapter 2

Background

2.1 DMC Working Principle and Implementation

The working principle of a digital microfluidic chip is the electrowetting effect. Electrowetting on dielectric (EWD) is the phenomenon that an electric field can modify the wetting behavior of a polarizable and/or conductive liquid droplet in contact with a hydrophobic, insulated electrode [2]. The application of a voltage between the liquid and the electrode results in an electric field through the insulator that lowers the interfacial tension between the liquid and the insulator surface according to the Lippman-Young equation [2]. Fig. 2.1 shows an example of the electrowetting effect.

Fig. 2.1: Electrowetting-on-dielectric (EWD) effect [2]
The droplet shown in Fig. 2.1 (left) is initially placed on the top of a hydrophobic insulated electrode without applying any voltage. In this case the contact angle is around 40 degree. Then a 70V voltage is applied to the droplet (usually the voltage is between 20 V-80 V) as shown in Fig. 2.1 (right). The top of the droplet is seen as a VDD, and the insulated electrode is seen as a GND. When the voltage is applied, the tension of solid liquid interfacial will be reduced. So the contact angle is increased and the contacted surface between the droplet and the insulated electrode is also increased. In order to implement the EWD effect, the digital microfluidic chip is mainly implemented as Fig. 2.2 [2]. This Figure gives the side-view of a DMC with a conductive glass top plate (GND), a fluid layer, and a bottom place with controllable electrodes (VDD). By applying a high voltage to the right (left) electrode of the central one, the droplet shown in Fig. 2.2 can be actuated to the right (left) cell.

Fig. 2.2: Working principle and structure of a DMC [2]
In Fig. 2.3, it shows the top view of a digital microfluidic array. We can see it is a 7 X 8 array and control pads are connected to the electrodes. Droplets can route and perform many different operations on the array by controlling the applied voltage to all electrodes.

### 2.2 DMC Operations

According to the structure of digital microfluidic chips and the electrowetting effect, operations can be conducted by changing the applied voltage to the electrodes. In this section, operations such as transport, mixing, split and dispensing of electrowetting-based digital microfluidic chips are described in details in the following discussions.
Transport: Before we move forward to other operations, we must understand how a droplet can move/transport on the digital microfluidic array. Droplet transport is operated over contiguous electrodes conducting different fluidic operations on the chip. A basic EWD device is based on charge control manipulation at the insulator interface of discrete droplets by applying voltage to control electrodes [2]. The device exhibits bilateral transport, is electrically isolated, uses a gate electrode for charge-controlled transport, has a threshold voltage, and is a square-law device in the relation between droplet velocity and gate actuation voltage [2]. Thus, the EWD device is analogous to the metaloxide-semiconductor (MOS) field-effect transistor (FET), not only as a charge-controlled device, but also as a universal switching element [3]. Another important factor to transport a droplet from one electrode to another is that the droplet must have overlap on both of the contiguous electrodes. Otherwise, no matter how high the voltage is applied to the electrodes, the droplet cannot move. The basic procedure for droplet transporting can be illustrated in Fig. 2.4. In Fig. 2.4 (a), we can see the droplet stays at the top of the left electrode, which is applied a VDD, in the meantime the right electrode is applied 0 V. Then both electrodes are applied VDD in Fig. 2.4 (b), and the contact angle of the droplet is reduced. The droplet has overlap between the left and right electrode. Eventually, the voltage for the left electrode is cut off, and the droplet successfully moves to the top of right electrode as shown in Fig. 2.4 (c).
Fig. 2.4: Behavior of droplet transportation

- **I/O and Dispensing:** Once we clear the concept of a droplet’s transportation, we explain the idea about how to load a droplet onto the digital microfluidics chip. Typically, we can directly pipette the droplet onto the digital microfluidic chip by using a syringe through a hole which is located at the boundary of the digital microfluidics array. Recently, designers have integrated reservoirs around the digital microfluidic array boundary and provided a continuous external supply source that keeps the on-chip reservoirs full. Then, droplets can be dispensed from the reservoirs to the digital microfluidics array. The procedure of dispensing is shown in Fig. 2.5 where three electrodes are connected to VDD one after another as shown in Fig. 2.5 (1, 2, and 3). Finally, we can apply high
voltage to the electrode of the reservoir and cut off the voltage of both electrodes between the reservoir and the array as shown in Fig. 2.5 (5). As a result, the droplet has been extracted from the reservoir into the digital microfluidics array as illustrated in Fig. 2.5 (6).

![Diagram of droplet dispensing phenomenon](image)

**Fig. 2.5: Droplet dispensing phenomenon**

- **Split/Merge:** Perhaps, the simplest but not the least useful fluidic operations in a digital microfluidic chip are the splitting of a droplet and the merging of two droplets into one. Procedure of splitting is shown in Fig. 2.6 and Fig. 2.7. During splitting, three electrodes are involved; the outer two electrodes are turned on and the contact angle is reduced, resulting in an increase of the overlap area between the droplet and the outer electrodes.
With the inner electrode off, the droplet expands to both outer electrodes which are applied voltage VDD. Thus, the splitting process is underway as the liquid forms a neck [2]. In general, the hydrophilic forces induced by the two outer electrodes stretch the droplet while the hydrophobic forces in the center pinch off the liquid into two daughter droplets [2].

Fig. 2.6: Side view of split operation

Fig. 2.7: Top view of split operation
The merge procedure is the opposite behavior of split. Two separate droplets can be joined together into a bigger one droplet referred to as merge. Fig. 2.8 shows the top view of how the merge operation works.

![Fig. 2.8: Top view of merge operation](image)

- **Mixing**: Mixing is one of the most important operations in digital microfluidic chips. It consists of merging and routing operations. Two or more droplets are merged within a cluster of cells which is called mixer and then routed to move around some pivots in the mixer as shown in Fig. 2.9.

![Fig. 2.9: (a) pivots and (b) routing path for a 2×4 microfluidic mixer](image)
Based on the operations discussed above, digital microfluidic chips can achieve the following functions and applications: (1) sample dilution and purification, (2) molecular separation, (3) assays, (4) PCR, and (5) sequencing of DNA [2].

2.3 Related Prior Work

<table>
<thead>
<tr>
<th>Physical Fault</th>
<th>Fault performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short between the droplet and the electrode.</td>
<td>A droplet is stuck at a faulty cell and cannot move further.</td>
</tr>
<tr>
<td>Electrode short due to metal connection between</td>
<td>A droplet is stuck between the two shorted electrodes and further movement of the</td>
</tr>
<tr>
<td>two adjacent electrodes.</td>
<td>droplet cannot be achieved.</td>
</tr>
<tr>
<td>Open between the electrode and the control source.</td>
<td>An electrode cannot be activated (stuck at fault).</td>
</tr>
<tr>
<td>Several conditions, such as particle contamination.</td>
<td>A droplet cannot move across the obstacle under this fault condition.</td>
</tr>
</tbody>
</table>

Table 2.1: Physical defects summary [8-16]

Although research in the design of digital microfluidics-based biochips has made a rapid progress in recent years [24-28], to the best of our knowledge, physical defects shown in Table 2.1 [8-16] developed by now are not complete, and no high level fault models that can reflect the behavior of many other defects have been developed. A few researchers have reported work on digital microfluidic chips’ testing [6-21] as shown in Table 2.2, and very few researches focus on the BIST circuit design of digital microfluidics chips. The first attempt to define
physical faults and to develop a test methodology for droplet-based microelectrofluidic systems is illustrated in [6]. It presents some physical structure faults which are really simple and shows how faults can be detected by electrostatically controlling and tracking droplet motion. Test planning and test resource optimization for droplet-based microfluidic arrays is investigated in [8]. It presents a heuristic approach to optimize the NP-hard planning problem.

<table>
<thead>
<tr>
<th>Test strategy</th>
<th>Description</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamiltonian path testing [8]</td>
<td>Moving droplet to every cell once following Hamiltonian path.</td>
<td>Missing detect some faults, such as electrodes short fault between two adjacent cells.</td>
</tr>
<tr>
<td>Euler path testing [13, 14]</td>
<td>Model the digital microfluidic array under test as an undirected graph, and then “eulerize” this graph [14].</td>
<td>Time consuming and miss-detect some faults.</td>
</tr>
<tr>
<td>Parallel scan like testing [15]</td>
<td>Load droplets to odd or even rows and columns and route them from one side to another side concurrently.</td>
<td>Incomplete fault models, flooding problem, miss-detect some faults.</td>
</tr>
<tr>
<td>Efficient parallel testing [16]</td>
<td>Parallel scan with chip partitioning into several test zones and retrieving the droplet back to inlet.</td>
<td>Incomplete fault models, partition un-thoroughly, miss-detect some faults.</td>
</tr>
</tbody>
</table>

Table 2.2: Classic test strategies

A concurrent testing methodology is proposed in [9], which is used to investigate the related problems of test planning and resource optimization. A defect tolerance methodology for digital microfluidic chip testing is published in [11]. In [13] and [14], a defect-oriented testing and diagnosis strategy is presented. It introduces some detailed structure fault models and proposes an Euler path algorithm based testing method and a binary search based...
diagnosis method. However, the physical fault models used do not cover all kinds of droplet fault behavior; these kinds of test strategies are timing-consuming and are not feasible in real chip testing, because they do not retrieve the droplet (which is stuck at the faulty cell) back to the reservoir that causes horrible flooding problems. Also for large size arrays, this kind of testing methods will consume enormous test time which is not doable in mass production.

In order to reduce the off-line test time and increase the fault coverage, a scan-like parallel testing and diagnosis method for digital microfluidic biochips is published in [15]. It develops more fault models, which are used for most of the testing strategy researchers, but are still not complete enough due to the inherited drawbacks of physical fault models. The scan-like parallel testing method works in such a way that all droplets routing from east (south) to west (north) at the same time, which reduces the time complexity to $O(n)$. If there is no droplet appearing in an outlet, we say there is a faulty cell along this path. However, this method just gives the test strategy without detailed test algorithm, and there are possibilities for misdiagnosis and flooding.

An efficient parallel testing and diagnosis strategy is proposed recently in [16], where several parts of a DMC are tested simultaneously to reduce test time, and droplets are moved back in the marching routes for diagnosis and avoiding grid flooding. However, their fault models are mostly inherited from [15]; simply mark a cell as faulty without telling which droplet marching direction is not functional from this cell. For example, it has been assumed in [16] and all of the fault models proposed so far in this field that if a droplet can successfully
move from direction i to direction j, then it can also successfully move back from direction j to
direction i. In fact, this is not true and thus traditional test and diagnosis strategies cannot
guarantee to produce fault-free digital microfluidic chips.
Chapter 3

Fault Modeling

Physical defects in digital microfluidics can be very complex and hard to analyze. There are so many different physical defects (e.g., broken wires, broken comb electrodes, short-circuit between electrodes, field coupling, holes, particles, etc.), and each defect may lead to different or the same influence in the microfluidic behavior. It is not possible to exhaustively consider all these different physical defects one by one. Even though we can develop many kinds of physical fault models, it is very inefficient to generate test patterns for each of those fault models. Therefore, we need to develop high (abstract) level fault models for digital microfluidic chips, such that as many faults as possible can be detected in a limited number of fault models. A high level fault model is based on the behavior of a droplet, and does not focus on the real defect of each particular component such as electrodes broken. Three major different types of high-level fault models are developed in this thesis, which are $FX$ fault, $PFX$ fault and $PFX_y$ fault.
3.1 FX Fault Model

3.1.1 Fundamental Concept

FX fault model is defined as a fault model where the droplet cannot fully move to its neighboring cell, and gets stuck at its original location. Letter ‘F’ stands for ‘fail’ and letter ‘X’ means one of the four directions: east, west, south, and north. Thus, FX means a fault fails to move to direction X. For example, ‘FE’ means the droplet is stuck at its original location and fails to transport to its east cell. Fig. 3.1 illustrates such a fault behavior. Further, ‘FW’, ‘FN’ and ‘FS’ are also described in Fig. 3.1. Physical defects causing this fault model can be many such as open circuit, broken wires, and broken comb electrodes. In this work, we extract the FX fault model to express all different physical faults which result in the same faulty behavior, and it is much easier to generate test patterns and greatly increase the fault coverage.

Fig. 3.1: FX fault model
3.1.2 Fault Analysis

If each cell on a microfluidic circuit can perform move operations to all four directions (move-to-north, move-to-south, move-to-east, move-to-west), then all the cells in this digital microfluidic chip are FX fault-free.

The notation of a FX fault is: \((i, j, FX)\), where \((i, j)\) is the coordinate of the cell being discussed (‘i’ is for row, and ‘j’ is for column) and FX is one of the four fault behaviors. For example, fault \((3, 5, FS)\) means once a droplet is in cell \((3, 5)\), it will fail to move north. But, droplet in cell \((3, 6)\) may still be able to move south into cell \((3, 5)\). With this fault model, for a \(N\times N\) microfluidic chip, it possibly has \((4N^2-4N)\) faults in total. This can be explained as follows. Each cell has 4 faults (FN, FS, FE, FW) and we have \(N^2\) cells, so the total number of faults is \(4N^2\). However, for the boundary cells along all four sides, each cell does not have the fault toward the outside, so the total number \(4N^2\) should be decreased by \(4N\) faults.

Previously, researchers thought a droplet can move from one cell to a direction can also move back from that direction. That is, \((i, j, FE) / (i, j, FS)\) and \((i, j+1, FW) / (i+1, j, FN)\) are thought as the same faults and can be detected by the same test pattern which can dramatically reduce the fault number. In fact, this is unfortunately not correct. Fig. 3.2 explains this reason which is also one of the major reasons for us to develop the FX fault model. From the figure, we can see the droplet can easily transport from the bottom cell to the top cell (Fig. 3.2(b)), but fails to move from the top to bottom (Fig. 3.2(c)) due to the tooth missing of the bottom cell (Fig. 3.2(a)). Thus, the total number of faults should be: \((4N^2-4N)\).
Among the \((4N^2-4N)\) faults, in reality, we may have any number of faults occurring simultaneously. That is, in the real chip operation, we may have two or more defects. Totally we may have \(2^{(4N^2-4N)}-1\) different faults if multiple FX faults may occur. The reason comes from that each fault among \((4N^2-4N)\) different faults can occur or not (i.e., two choices). Thus, the number of all combinations of multiple FX faulty cases is \(2^{(4N^2-4N)}-1\), where “-1” represents the fault-free case.
3.1.3 Experimental Support

The experiment platform is a $4 \times 4$ digital microfluidic array (Fig. 3.3 [22]). Totally, 16 cells are connected to 16 control pads. The droplet is dispensed from a reservoir located at the right bottom corner which is also used to collect the output droplet.

Fig. 3.3: FX fault model verification [22]

In order to verify the FS fault model using a physical defect, a droplet was dispensed from the reservoir and trying to move to the south cell, but it was stuck at its original cell and failed to move to the south cell— we call it Fail-to South. The missing tooth in the south cell (Fig. 3.2(a)) is simulated by disconnect the control signal of the south cell. We were not trying to do the experiment for FN, FE and FW one by one, since they are all logically the same as the FS fault model.
3.2 PFX Fault Model

3.2.1 Fundamental Concept

PFX fault model is defined as a fault where the droplet cannot fully move to its neighbor cell, but can partially move to the target location and is stuck at the conjunction of these two connected cells. The letter ‘P’ stands for ‘partially’, so PFX means partially-fail-to-X. For example, ‘PFE’ means the droplet is trying to transport to the neighboring cell of its east side; however the droplet can only partially moves to that target cell, and then it is stuck in the middle of the original cell and its east neighboring cell. Fig. 3.4(a) illustrates such faulty behavior, and ‘PFW’, ‘PFN’ and ‘PFS’ are also described in Fig. 3.4.

![Fig. 3.4: PFX fault model](image-url)
3.2.2 Fault Analysis

A digital microfluidic chip is said to be PFX fault free, if there is no droplet stuck at the middle of any pair of two neighboring cells. The notation of a PFX fault in a digital microfluidic chip is: \((i, j, PFX)\). Similar to the notation described in the FX fault model, \((i, j)\) is the coordinate of the cell being discussed and PFX is one of the four fault behaviors. For example, fault \((3, 4, \text{PFS})\) means once a droplet is on cell \((3, 4)\), it will partially fail to move south and is stuck in the middle of cell \((3, 4)\) and cell \((3, 3)\).

Here, a new concept called fault equivalence must be introduced. Some faults might have different fault names, but indicate exactly the same fault. They are equivalent. Therefore, in the above example, fault \((3, 4, \text{PFS})\) is exactly the same as fault \((3, 3, \text{PFN})\). We only need to consider one of them for test pattern generation. By considering fault equivalence for the PFX fault model, for a \(N \times N\) microfluidic chip, it does not have as many faults as FX fault model does (i.e., \(4N^2-4N\) faults). Instead, it has only \(2N^2-2N\) PFX faults. Similar to the analysis in the FS fault model, the total number of faults is \((4N^2-4N)\) if fault equivalence relationship is not considered. However, since every two PFX faults sharing the same edge (e.g., \((3, 4, \text{PFS})\) and \((3, 3, \text{PFN})\)) are equivalent faults, and so the total number of PFX faults is \((4N^2-4N)/2=2N^2-2N\).

Among the \((2N^2-2N)\) single PFX faults, in reality, we may have any number of PFX faults occurring simultaneously, similar to FX faults. In this case, totally we have \(2^{(2N^2-2N)}-1\) different multiple PFX faults and the reason can be discussed as above for multiple PF faults.
3.2.3 Experimental Support

A 4 X 4 digital microfluidic chip is also used to verify the PFX fault model as illustrated in Fig. 3.5. The droplet was trying to move to the south neighboring cell, but it got stuck at the middle of these two cells as shown in Fig. 3.5(right) [22], which illustrated the PFS fault. The physical fault can be simulated by injecting a short circuit between the electrodes of these two cells as shown in Fig. 3.6 (e.g., apply voltage to two cells concurrently). Obviously, there are also many other physical defects that can result in such a faulty behavior.

Fig. 3.5: PFS fault model experiment [22]

Fig. 3.6: Physical fault injection for PFX fault
3.3 PFXy Fault Modeling

3.3.1 Fundamental Concept

*PFXy fault* is short for partially-fail-to-Xy (directions) fault. It has a similar part with PFX in that a droplet only partially moves to the target cell. But for the PFXy fault model, a droplet not only partially moves to the target cell, but also moves to another cell which has a diagonal relationship with the target cell on the chip. Thus, letter ‘X’ in this fault model represents the direction the droplet trying to move to and letter ‘y’ represents the direction which the droplet should not move to. For example, ‘PFEn’ means the droplet is to transport to the neighboring cell of its east side, however the droplet split into its east cell and its north cell. Fig. 3.7 illustrates such a fault behavior. PFXy for the other directions can be discussed similarly.

![Fig. 3.7: PFEn fault model](image-url)
3.3.2 Fault Analysis

The notation of PFX fault in a digital microfluidic chip is: \((i, j, PFXy)\). Similar to the notation described in the above fault model, \((i, j)\) is the coordinate of the cell being discussed and \(PFXy\) is one of the eight fault behaviors (PFEn, PFEs, PFWn, PFSw, PFSe, PFNe and PFNw). For instance, fault \((3, 4, PFNe)\) means that a droplet is in cell \((3, 4)\), and it tries to move to north but fails. Instead, it splits into cell \((3, 5)\) and cell \((2, 4)\) and they are still connected.

Although every cell has eight different faults, four of them are equivalent to the other four faults. For example, PFEn is equivalent to PFNe, since the droplet is stuck at the same location which is at the diagonal of the east and north cells. Thus, after fault collapsing every cell has four different kinds of PFXy faulty behavior. In this case, for a \(N\times N\) microfluidic chip, totally there are \(2(N-1)^2\) PFXy faults, because a \(N \times N\) array has \((N-1) \times (N-1)\) crossing sites and each crossing sites has two faulty behaviors. For example, in Fig. 3.8, it is a 4 X 4 array. It has \((4-1) \times (4-1) = 9\) crossing points and each point has two types of PFXy faults. So, totally this array has \(9 \times 2 = 18\) PFXy faults. According to the above analysis, we can easily come to the conclusion as we did in the above section that among the \(2N^2 - 4N + 2\) PFXy faults, totally we may have \(2^{(2N^2 - 4N + 2) - 1}\) different multiple PFXy faults.
3.3.3 Experimental Support

A $4 \times 4$ digital microfluidic chip is also used to verify the PFXy fault model as illustrated in Fig. 3.9 [22]. Here, the droplet is moving toward its west neighboring cell; however, due to the PFWn fault, the droplet fails to reach its target cell. Instead, it splits into its west and north cells, but still connected. This experiment was conducted by making these two cells connected electronically at the same time resulting in a short effect as shown in Fig. 3.10.
Fig. 3.10: Physical fault injection for PFWn

Short circuit between two diagonal teeth
Chapter 4

Test Strategy

The purpose of a test strategy for digital microfluidics chips is to find out an efficient way to sensitize and propagate all faults located in each array to its outputs for observation by generating the smallest number of test patterns. In digital VLSI circuits, the test pattern for a certain stuck-at fault is digital inputs. However, in microfluidic chips, the test pattern for a certain cell fault is a route. The route delivers a microfluidic droplet from a certain inlet to this cell under test (CUT), activates it toward the faulty movement from this cell (i.e., fault activation), and then delivers it to a certain outlet for detection (i.e., fault detection). If this droplet is detected at the outlet, then the suspected fault does not exist. However, if the droplet does not show up at the outlet, then the fault does exist and it is detected.

As test engineers, our task is to find out a minimum set of routes (paths) to cover (detect) all the faults in the chip. Furthermore, this minimum set of routes can work simultaneously during testing to reduce the testing time (i.e., by parallel testing [15]). For each path, it may test different faults simultaneously. Theoretically, all faults along this path can be detected. For
example, in Fig. 4.1, for the path marked as red, it can simultaneously detect all the following faults along it: (1, 2, FE/PFE/PFEn/PFEs), (2, 2, FE/PFE/PFEn/PFEs), (3, 2, FE/PFE/PFEn/PFEs), (4, 2, FN/PFN/PFNe/PFNw), (4, 3, FE/PFE/PFEn/PFEs), (5, 3, FE/PFE/PFEn/PFEs), (6, 3, FN/PFN/PFNe/PFNw), (6, 4, FE/PFE/PFEn/PFEs), (7, 4, FE/PFE/PFEn/PFEs).

![Fig. 4.1: A test route](image)

We do not have to test each path one after another, because this wastes too much test time. Instead, we need to work out an effective algorithm to schedule all paths to be exercised so that they will not conflict with each other. That is, different paths are not allowed to pass through a common cell at the same time; at any moment, they must maintain a minimum gap (one cell gap) between each other to avoid droplet interference. Thus, March-p/March-p+ and March-d algorithms are developed to generate such efficient test patterns and to achieve the fault type distinguishing respectively for digital micrfluidics chips.
4.1 March-p Test

4.1.1 Test Pattern Generation

According to all three major high level fault models discussed in Chapter 3, the march-p (parallel testing) algorithm is developed to generate simple and efficient test patterns to test the digital microfluidic chip. The algorithm is described in Fig. 4.2, and the corresponding example for a $8 \times 8$ arrays is shown in Fig. 4.3.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Route droplets from north (inlet) to south (output) at odd columns simultaneously.</td>
</tr>
<tr>
<td>2</td>
<td>Route droplets from south (inlet) to north (output) at odd columns simultaneously.</td>
</tr>
<tr>
<td>3</td>
<td>Route droplets from north (inlet) to south (output) at even columns simultaneously.</td>
</tr>
<tr>
<td>4</td>
<td>Route droplets from south (inlet) to north (output) at even columns simultaneously.</td>
</tr>
<tr>
<td>5</td>
<td>Route droplets from west (inlet) to east (output) at odd rows simultaneously.</td>
</tr>
<tr>
<td>6</td>
<td>Route droplets from east (inlet) to west (output) at odd rows simultaneously.</td>
</tr>
<tr>
<td>7</td>
<td>Route droplets from west (inlet) to east (output) at even rows simultaneously.</td>
</tr>
<tr>
<td>8</td>
<td>Route droplets from east (inlet) to west (output) at even rows simultaneously.</td>
</tr>
</tbody>
</table>

Fig. 4.2: March-p algorithm description

The march elements are extracted to represent the march-p algorithm, and the complete march algorithm notation is given in Table 4.1. In Table 4.1 there are eight march elements, denoted as M0, M1…M7, which correspond to step 1 to step 8 in Fig. 4.2. The arrows
represent four different droplet directions. For example, \(\downarrow\) means droplet routes from north to south. ‘O’ (‘E’) represents ‘Odd’ (‘Even’), which gives odd (even) rows or columns. The time complexity of the march-p algorithm is \(O(8N)\), since we have to route the droplets from one side to another side of the array in 8 times and assume it is a two dimensional \(N \times N\) digital microfluidics array.

\[
\downarrow(O); \uparrow(O); \downarrow(E); \uparrow(E); \Rightarrow(O); \Leftarrow(O); \Rightarrow(E); \Leftarrow(E);
\]

Table 4.1: The march-p algorithm

Fig. 4.3: Illustration of march-p algorithm
In Fig. 4.3, there are sources and sinks around the digital microfluidic chip; four droplets are loaded from the sources to odd columns at the same time; in step 3, we have to load the droplets from the source to even columns one more time, then continue to do the following steps. At the end of steps 4 and 8, we observe whether the droplets are at the outlets of the chip through a microscope or an integrated capacitive sensing circuit [15]. In order to overcome the flooding issue, which is an important problem in previous test methodologies, we have to blow away and clear the droplets after each test steps, and then load new droplets for next steps.

4.1.2 Test Analysis

In [15] and [16], similar scan-like parallel test patterns have been used, but some key scan patterns such as step 2, 4, 6, and 8 are missed. This is because they simply assumed that a droplet can route from east/north to west/south during testing can also move back, and vice versa. In fact, it is not true. We already explained this in Chapter 3. Thus, the fault coverage in [15] and [16] is far smaller than our march-p algorithm. The march-p algorithm can successfully cover 100% of all fault cases based on the above three major high level fault models. We can illustrate this conclusion by the following three examples:
1. If we can detect all loaded droplets at the correct outlet locations, then we conclude that the chip is FX fault-free. For example, for a given N x N two dimensional digital microfluidic array, we randomly select any cell (i, j) for testing as shown in Fig. 4.4. If the droplet can successfully route through (i, j) following the paths defined by the march-p algorithm, then from Fig. 4.4 we know that there are no FE, FW, FN and FS faults at cell (i, j). According to this, we can say that all cells in the chip have no FX faults and the chip is FX fault-free if all droplets appear at the corresponded outlets after testing. We can also use fault simulation for march-p algorithm test patterns to reach the same conclusion.

![Fig. 4.4: Cell (i, j)](image1)

![Fig. 4.5: Example of PFX fault](image2)
2. If we can observe all the loaded droplets at the right outlet, then the chip is PFX fault free. For example, if the cell has a PFX fault, the droplet will be stuck at the middle of itself and its target cell. Due to the fact that there is no overlap between the stuck droplet and the next cell of its target cell, the droplet cannot further move to the outlet (see Fig. 4.5). Therefore, the fault effect is the same as a FX fault.

3. If we can observe all the loaded droplets at the designated outlets, then the chip is PFXy fault-free. For example, if the cell has a PFXy fault between cell (2, 3) and (3, 2) as shown in Fig. 4.6, the droplet moving from cell (3, 3) to cell (2, 3) will split to its diagonal cell (3, 2). For the target cell (2, 3), there is no overlap between it and its next cell (1, 3). So the droplet cannot further move along with its defined path to the outlet, and the fault effect is also the same as a FX fault.

![Fig. 4.6: Example of PFXy fault](image_url)
After going through the above examples, we can come to the conclusion (without a formal proof) that the march-p algorithm can test all three types’ high level faults discussed in chapter 3.

### 4.2 March-p+ Test

Although the time complexity of march-p is linear, it is not efficient (fast) enough for large size chips. In order to further increase the test speed, we revise the march-p algorithm by scheduling some of the steps executed simultaneously, and generate an advanced parallel testing strategy named march-p+ test as shown in Fig. 4.7 and Fig. 4.8.

1. **Step 1:** At \( t=0 \), route droplets from north to cells \( (i, \lfloor N/2 \rfloor -1) \) and route back to the sources in odd columns \( i \);
2. **Step 2:** At \( t=4 \), route droplets from south to cells \( (i, \lfloor N/2 \rfloor -1) \) and route back to sources in odd columns \( i \);
3. **Step 3:** Redo step 1 and step 2 in even columns;
4. **Step 4:** At \( t=0 \), route droplets from west to cells \( (\lfloor N/2 \rfloor -1, j) \) and route back to sources in odd rows \( j \);
5. **Step 5:** At \( t=4 \), route droplets from east to cells \( (\lfloor N/2 \rfloor -1, j) \) and route back to sources in odd rows \( j \);
6. **Step 6:** Redo step 4 and step 5 in even rows.

![Fig. 4.7: March-p+ algorithm](image)

We schedule patterns from north/south to south/north and east/west to west/east simultaneously with 4 time units (route steps) difference to avoid both side droplets meeting at the middle of the array. It can be easily verified that march-p+ has exactly the same fault
coverage as march-p, but the test time is reduced by 50% when compared with march-p.

4.3 March-d Test

4.3.1 Test Pattern Generation

March-p and march-p+ test methodologies focus on routing the droplets in horizontal and vertical directions. It is interesting to see whether the droplets routed in diagonal direction as shown in Fig. 4.9 can detect and distinguish faults. Every cell around the boundary of a digital microfluidic chip can be an inlet, outlet, both or a regular cell. In order to figure out whether the chip is fault-free, every cell must be routed by droplets in all directions. The droplet routing effect using all diagonal paths developed later is shown as Fig. 4.10.
Fig. 4.9: Diagonal paths

Fig. 4.10: Paths covered by all diagonal paths
We find from Fig. 4.10 that all cells except the boundary cells can be routed with four directions which is the same as the effect of March-p test patterns. The four corners are not routed by any path and the vertical (horizontal) boundary cells are not routed by droplets from their north and south (east and west) cells. Therefore, in order to test all the cells, we must test the boundary cells first as shown in Fig. 4.11 before we route the inner cells. Again, we emphasize that boundary cells are not completely tested due to the march-d algorithm that will be presented later.

![Outer cell routing](image)

Fig. 4.11: Outer cell routing

There are two inlet/outlet cells in the example shown in Fig. 4.11. Two droplets are loaded from inlet/outlet cells (blue cells in Fig. 4.11), routing along the red path to another blue cell (inlet/outlet), and then do the opposite routing procedure for each path testing. In order to decrease the boundary test routing time, we can double the number of inlet/outlet cells (i.e., every boundary side has one inlet/outlet cell), which can half the routing distance.
Due to some special properties of diagonal routing, we cannot simply schedule all the paths that are in the same directions to route at the same time, though there is one empty cell between them. For example, we cannot schedule all the droplets in even columns/rows cells to route from southwest to northeast simultaneously as shown in Fig. 4.12, because this will result in errors during test (both paths interfere and mix together).

We can observe that droplets ‘a’ and ‘b’ are loaded at the even rows (Fig. 4.12). By march-p or march-p+ test, they can be scheduled to route at the same time without any problem. If droplets ‘a’ and ‘b’ follow the diagonal path, both work properly when they route to their start positions in Fig. 4.12. However, in the next step, a and b both need to move to their north (top) cells—cell 1 and cell 2, and then problem occurs, this is because cell 1 and cell 2 will be applied high voltage at the same time, and droplet ‘a’ has overlap not only with cell 1 but also with cell 2, which makes droplet ‘a’ split into cell 1 and cell 2. In addition,
droplet ‘b’ moves to cell 2 and mix with part of droplet 1. The test result is thus incorrect when this phenomenon occurs. Thus, developing an efficient and bug-free test methodology for diagonal scan is necessary. With this motivation, an algorithm called march-d is presented in Fig. 4.13.

Outer cells route:

Step: route and test the boundary cells. (Fig. 4.11)

Inner cells route:

Step 1: at T=0, route the longest path (start from cell (1, 2)) and each path which are two cells away from it concurrently from SW to NE as show in Fig. 4.14(a).

Step 2: at T=5, route and test the next neighbor cells with step 1 concurrently (Fig. 4.14(b)).

Step 3: at T=10, route and test the rest cells except path that start from cell (3, 1) with step 1 and step 2 concurrently (Fig. 4.14 (c)); at T=11, route the last path which is start from cell (3, 1) (Fig. 4.14(d)).

Step 4: after all the above steps, redo step 1 to step 3 in opposite direction.

Fig. 4.13: March-d algorithm
Fig. 4.14: Illustration of march-d test

Fig. 4.15: Illustration of test time
With the above scheduling constrains, we split the routing procedure into three stages that can guarantee no pseudo-fault occurrence during the entire testing procedure with minimum test time. The test time depends on two factors: route steps (elapsing time/path length) and route start time. The test time for finishing routing a test path is the sum of the route steps (path length) and route start time (e.g., T=5). For march-d algorithm, the equation of a path length is:

\[ N-i+N-i-1=2(N-i)-1 \]

where N is the dimension of a chip and i is the difference between the location of a path starting cell and cell (1, 1). For example, the value of I for the blue cell in Fig. 4.15 is 4-1=3. The test time of the blue path in Fig. 4.15 is: path steps + start time = 2(8-3)-1+5=9+5=14, and the test time of green path is 7+0=7. Both match the steps we count directly from the Figure.

Further, the overall test time of an entire chip is determined by the test time of the path which has the longest finishing time. For our march-d algorithm applied to Fig. 4.14, the last finished path is the yellow path whose test time is \(2(N-i)-1+start\) time\(=2(N-2)-1+12=2N+7\), which also determines the entire inner loop test time of the whole chip. The reason why start the yellow path from cell (3, 1) at T=12 rather than T=10 as shown in Fig. 4.14 is that at T = 10 another test path starting from cell (1, 3) is also active and it may interfere the path starting from cell (3, 1). Therefore, for safety reason we delay two time unit for the path started from cell (3, 1). The time complexity for the march-d algorithm is \(T \text{ (outer route)} + T \text{ (inner route)} = 4N + 2*(2N+7) = 8N+14\). By adding two more inlets/outlets for the outer route, the time
complexity can be reduced to $6N+14$.

### 4.3.2 Test Analysis

From Fig 4.16 we can see our march-d algorithm can detect all FX and PFX faults. We need further investigate the characteristics of PFXy faults to discuss whether march-d can detect all PFXy faults. In Fig. 4.17, assuming cell (2, 2) has a PFNe Fault, the droplet will split into cell (3, 2) and cell (2, 3) but is still connected as shown in Fig. 4.17 (b). However, the droplet can successfully move to cell (3, 3) and eventually reach the outlet when the diagonal test pattern is applied. The reason is when the droplet splits into cell (3, 2) and cell (2, 3), both smaller droplets still have overlap area with cell (3, 3). Once cell (3, 3) is applied a high voltage, both smaller droplets will still move to cell (3, 3). Instead, if the droplet shown in Fig. 4.17 (a) is trying to move to cell (2, 4) rather than (3, 3), it fails, since there is no overlap area between the droplet and cell (2, 4).
In order to verify our analysis in Fig. 4.17, a 4 x 4 digital microfluidic chip is utilized to conduct the fault simulation experiment as shown in Fig. 4.18 [22]. The droplet is following the red path which is a diagonal one. The cell in (a) has a PFWn fault and we have observed that the droplet failed to move to its west cell and split into its west and north neighboring cells, but was still connected Fig. 4.18 (b-d). In the next step, cell (2, 4) was applied a high voltage and the droplet successfully moved to cell (2, 4) that was not influenced by the PFXy fault.

Therefore, the march-d algorithm is guaranteed to detect 100% FX faults and PFX faults with a high test speed, but cannot detect PFXy Faults. Although fault coverage of the march-d algorithm is smaller than the march-p and march-p+ algorithms, it is very important to implement the fault type distinguishing test methodology, which will be discussed in section 4.4.
Fig. 4.18: Fault simulation experiment for PFWn under march-d [22]

4.4 Fault Type Distinguishing Test Strategy

4.4.1 Motivation

Only knowing whether a digital microfluidics chip is fault-free by testing is not adequate. If we can figure out the fault type during test without increasing too much test time, it is very helpful for further fault diagnosis and yield improvement.

The reason why identifying the fault type can greatly increase the fault location (debugging) accuracy is that a fault may be related to one fault location (e.g., FX fault model) or two (e.g., PFXy fault model). In [16], the stuck droplet is retrieved back to its source with
a binary search based methodology to identify the fault location. However, the test strategy in [16] cannot distinguish fault types, and thus its diagnosis method cannot find out how many cells are faulty (i.e., cannot identify the accurate fault location). Assume a droplet routes from the inlet to a PFXy faulty cell (blue cells) along any test paths in Fig. 4.19. Once the droplet gets stuck, it will be retrieved back to the inlet by the proposed binary search based diagnosis strategy in [16]. If we do not have idea of the faulty type, we cannot know how many cells are faulty at that location. If we know why the droplet gets stuck, we can easily infer the number of faulty cells in that location. For example, if the fault is FX, the faulty cell is one; if the fault is PFXy, two faulty cells are involved. Therefore, to develop a fault type distinguishing test strategy is necessary and very useful for defect diagnosis and fault repairing to increase the yield.

Fig. 4.19: Example of misdiagnosis
Another very important reason of knowing the fault type is to increase the manufacturing yield without fault repairing. This is because, for a particular chip, some operations and functions can still be executed with the existence of some faults. For example, if part of a chip has a PFXy fault, then the chip can still be used to conduct operations that allow routing in a diagonal path. Therefore, we do not get rid of every faulty chip which greatly increase the yield without repairing the defect. Finally, if we find a location with a specific fault type frequently, we can fix the manufacturing process to eliminate the defect.

With the above motivations, developing a fault type distinguishable test strategy is necessary and very important.

### 4.4.2 Test Pattern Generation for Distinguishing Fault Types

In Chapter 3, we have generated three high level fault models-FX, PFX and PFXy. Actually, fault models FX and PFX can be tested using the same set of test patterns, since the fault-related cells are along with the same direction of each test pattern. The fault distinguishing test patterns generated here can successfully distinguish PFXy fault from FX and PFX faults.

Fortunately, the test algorithms we developed before can be used to achieve our goal. The march-d algorithm can detect FX and PFX but not PFXy; the March-p and March-p+ algorithm can detect all of them as shown in Fig. 4.20. However, considering the test
efficiency, we cannot simply apply both algorithms sequentially. The algorithm of our proposed fault distinguishing test strategy is shown in Fig. 4.21, and is synthesized by optimizing part of march-p and march-d steps.

![Diagram of fault types](image)

**Fig. 4.20:** Fault type distinguishing by different algorithms

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply march-d algorithm.</td>
<td>Apply step 1, 3 of march-p.</td>
</tr>
</tbody>
</table>

**Fig. 4.21:** Fault type distinguishing test generation

The reason why we only apply two steps of march-p to test PFXy faults is that when step 1 is completed, all FX and PFX faults are detected, and only PFXy faults remaining to be detected. However, for PFXy faults, one direction scan pattern is enough to detect all of them. As shown in Fig. 4.22, we find that for a given PFXy fault, any one of the four direction scan patterns can active and detect this fault as the red arrow shows. Thus, we can reduce the test length from 4N to N. So, the total test time can still be tolerable, though we combine two test algorithms.
together. The time complexity is thus

\[
Time \text{ of March-}d + 2N = 8N+14.
\]

for an N x N two dimensional array. One more thing needs to mention is that we can replace march-p with march-p+ to further increase the test speed. We emphasize again that the overall test length is not too much longer than each march-d (6N+14) and march-p (8N), with the benefit of identifying fault types.

Fig. 4.22: One way detection for PFXy faults
4.4.3 Test Analysis

<table>
<thead>
<tr>
<th>Test algorithm</th>
<th>Error found</th>
<th>Error-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>March-d</td>
<td>FX and PFX faults</td>
<td>No FX and PFX faults</td>
</tr>
<tr>
<td>Steps 1 and 3 of March-p</td>
<td>PFXy faults</td>
<td>No PFXy faults</td>
</tr>
</tbody>
</table>

Table 4.2: Fault type identification analysis

From Table 4.2, if there is no error found after apply both march-d and steps 1 and 3 of march-p, we can conclude that the DMC does not have any FX, PFX, and PFXy fault. However, if an error is found after march-d testing, we can conclude that FX or PFX or both fault exist. Fault location by binary search using march-d can be performed immediately. If march-d does not find any errors, but march-p (steps 1 and 3) finds errors, we know that there is an PFXy fault, and fault location can be performed. Unfortunately, if march-d and march-p both find errors in the overlap area, fault type distinguishing fails. In this case, the chip may contain FX or PFX or both, with or without the existence of PFXy. Future research will be required to study how to handle this case.
4.5 Discussion

During digital microfluidics chip testing, many factors can affect the results and many issues might lower the quality of test strategies. In this section, we will discuss several testing issues and present some feasible solutions for them.

✧ **Droplet volume**

The result of our testing methodology could be affected by the size and volume of a certain loaded droplet. If the volume is too large or too small, it cannot move in a fault-free chip. If the droplet is larger than the required size, it may pass some faulty cases that result in fault missing. On the other hand, if the droplet is smaller than the required size, it may generate pseudo-faults that result in good chips killed by testing. Therefore, the correct way is to choose the droplet volume larger than the minimum droplet and smaller than the maximum droplet (customer required size) under normal working conditions to guarantee the test quality.

✧ **Flooding problem**

Flooding is an important issue in digital microfluidics chip testing research, which has a very negative influence on the quality of test result. No matter in [15, 16] or our fault type distinguishing test strategy, there always exists flooding phenomenon if there are droplets stuck inside the chip resulting from previous test patterns. For example, during march-p testing, if one cell has FS fault then the droplet will be stuck at the faulty cell as
shown in Fig. 4.23 (a). When we continue to apply the next pattern, the other droplet would be merged with the previous stuck droplet to form a bigger one, which will result in errors during testing.

![Fig. 4.23: Illustration of flooding issue](image)

Recently, researchers have developed some methodologies to avoid the flooding problem during testing such as retrieving the droplet back to inlet/resource [16]. But those ways do not really overcome the problem. For example, in Fig. 4.24 (a), if the droplet is stuck at a FS cell, we try to retrieve the stuck droplet back to the north inlet rather than apply the next test pattern. It is a good solution, if there is no FN fault in the previous passed cells. However, once there is a FN faulty cell in its retrieval path, the droplet will still get stuck in the chip and is impossible to move out. We know the possibility for this phenomenon to occur is quite high. In this thesis, we suggest a new methodology based on the structure of the digital microfluidics chip. We know that most
DMC devices are filled with oil as the medium for chemical droplet moving. It is pretty easy to fill in oil into the devices. Therefore, in our test platform, we can build a special syringe to fill the oil into the devices after applying every testing pattern. In this way, not only can we clean the stuck droplet, but also we clean the contaminations. This greatly improves the testing quality.

✧ **Design for Testability**

![Fig. 4.24: Illustration of cell testability](image)

Design for testability techniques have been very popular in current VLSI circuit design industry. It greatly increases the controllability and observability of VLSI circuit testing. For a digital microfluidic chip, it is somehow the same as a classroom. We can only load a droplet from one of the windows and observe the droplet from another window. We do not know what is going on inside the classroom. An efficient DFT
technique for DMC is that we build extra devices to control and observe the contents inside the classroom. By controlling and observing droplets in an efficient way can dramatically increase the test efficiency. Before we move to suggest a DFT methodology for DMC, we want to introduce two concepts for DMC first: *controllability* and *observability*.

For each cell $i$ in the chip, the controllability of $i$ is the number of cells a droplet needs to travel from the nearest inlet into this cell. The observability of $i$ is the number of cells a droplet needs to travel from cell $i$ to a nearest outlet for observation. For example, in Fig. 4.24, the controllability of A is 4 and the observability is 5. In fact if the inlets and outlets are around the chip boundaries, the sum of controllability and observability (called testability) is a
constant. We can do nothing to increase the testability without designing a new structure to
access more test points for the digital microfluidics chip. The new DMC structure in test
mode is shown in Fig. 4.25. In order to increase the testability of cells within the inner square,
we insert a loading port inside the chip. The way to determining the location of a loading port
depends on the partition of the entire test area. For example, in a test zone where the droplet
cannot be loaded and tested from an inlet, we must insert a load point for that area as well as a
capacitive sensing circuit for observing the test result. This is similar to inserting a test point in
the inner test zone in Fig. 4.25. In this way, the cells in the inner test zone have better testability,
since the distance to between each of them and inlets/outlets is shorter than before. Therefore,
we can apply parallel test patterns for that area to increase fault diagnosis resolution and the
test speed.
Chapter 5

Built-in Self-test Design and Emulation

5.1 Why Built-in Self-test

With advances in current EWD-based digital microfluidics chip manufacturing technology, scalable size and mass productivity will become major testing challenges. Traditional test techniques that generate and apply the test patterns manually have become quite expensive and cannot achieve high fault coverage for large size chips.

One approach to alleviate these testing problems is to incorporate built-in self-test (BIST) [29] features into its digital circuit controller when we design a DMC device. With BIST, circuits that generate test patterns and analyze output responses are embedded in the DMC device’s digital circuit part or elsewhere on the device’s motherboard.

There are two general categories of BIST techniques for testing VLSI random logic: (1) online BIST and (2) offline BIST as shown in Fig. 5.1[29]. Online BIST is performed when
A digital VLSI chip is in normal operational mode. It can be done either concurrently or nonconcurrently. In concurrent online BIST, testing is conducted simultaneously with normal functional operation. In nonconcurrent online BIST, testing is performed when the functional circuitry is in idle mode [29]. Offline BIST is performed when the functional circuitry is not in normal mode. The BIST developed for digital microfluidics chip in this chapter is a structural offline BIST.

5.2 Architecture Design of DMC with BIST

Fig. 5.2: Block diagram of a BIST system
The structure of a digital microfluidics chip with BIST design is shown in Fig. 5.2. The BIST circuit generates test patterns through its test pattern generator. TE is used to select test patterns or working (normal) inputs for the test pattern to address mapper. The test pattern to address mapper is a logic circuit that converts each test pattern to particular addresses for the DMC array. Since the voltage applied to digital microfluidics chips is very high (40-70 volts), we need voltage level shifters [30] to amplify the low voltage for every cell. Each integrated detector sends the response waveform to the response logic, which analyzes the result and sends a compressed signal (for square waveform, e.g.,111000) to the BIST circuit for comparing with the fault-free response.

Ideally, the BIST logic circuit can be divided into three parts: test pattern generator, comparator, and controller. March algorithms are implemented by the test pattern generator to automatically generate all test patterns. The comparator gets the test output result from the response logic, and then compares the result with the expected value (simulated fault-free result). If the output matches the expected value, the BIST circuit will output a signal (P: pass) shows that the chip is fault-free; otherwise it will output a faulty signal (F: fail). The $\text{CTLC}$ signal for the comparator is to control the test finish time, which is driven by a counter, since the total test time for a given device is fixed. The BIST controller takes charge of the overall BIST flow. In normal mode, the Start signal is set to low, and the BIST circuit will not be activated. During test mode, the BIST circuit is active by the Start signal and the test pattern generator works when it gets the command from controller through the $\text{CTLT}$ signal. After finishing the test procedure, TE is used to select normal inputs and the DMC system goes
back to normal mode and the DMC can start regular operation.

For recent research, capacitive sensing circuit has been widely used to detect the existence of a droplet output. A capacitive sensing circuit is connected to the corresponding electrode of a cell or sink. As we see in Fig. 5.3 [15], if there is no droplet in the target cell, the output is logic low. When the target cell has a droplet, the dynamic capacitance value will be changed which causes the output to generate logic high. As shown in Fig. 5.3, a series of periodic square waveform will be generated if logic high is generated. According to the value of the output, therefore, we can determine the occurrence of a droplet at the target cell.

![Capacitive sensing circuit](image)

Fig. 5.3: Capacitive sensing circuit [15]

### 5.3 Implementation and Emulation

In this thesis, instead of implementing a real BIST CMOS chip and digital microfluidic chip, we use FPGA to emulate the BIST circuit behavior and display the results on a virtual digital microfluidic array. The emulation platform is an ALTERA DE2 FPGA board [31] and a LCD monitor shown in Fig. 5.4.
5.3.1 Display Design

The DE2 board includes a 16-pin D-SUB connector for the VGA output. All VGA synchronization signals are provided directly from the Cyclone II FPGA, and an analog device ADV7123 triple 10-bit high-speed video DAC is used to produce the analog data signals (red, green, and blue) [31]. The resolution of displaying the two dimensional array is 640 * 480. The size of a cell is defined as 40 *40 with 2 unit gap with other cells and the color is assigned as blue.

Table 5.1 shows the Verilog code of generating the 8 X 8 two dimensional cell arrays (blue color). The size of droplet and defective cells is the same as that of regular cells. Droplets are represented as red and defective cells are represented as green. The Verilog code
of displaying a sample droplet and a sample defective cell are given in Table 5.2 and Table 5.3, respectively.

<table>
<thead>
<tr>
<th>8 x 8 cell array</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 5.1: Cells array design</strong></td>
</tr>
<tr>
<td>always@(pixel_row, pixel_col)</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>for (i=0;i&lt;8;i=i+1)</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>for (j=0;j&lt;8;j=j+1)</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>if (180+j<em>40+2</em>j &lt;= pixel_col + sq_size &amp;&amp; 180+j<em>40+2</em>j + sq_size &gt;= pixel_col &amp;&amp; 80+i<em>40+i</em>2 &lt;= pixel_row + sq_size &amp;&amp; 80+i<em>40+i</em>2 + sq_size &gt;= pixel_row )</td>
</tr>
<tr>
<td>sq[i][j]&lt;= 1;</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>sq[i][j]&lt;= 0;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defective cells</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 5.2: Defective cell design</strong></td>
</tr>
<tr>
<td>always@(pixel_row, pixel_col)</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>i=4; // faults location</td>
</tr>
<tr>
<td>j=4;</td>
</tr>
<tr>
<td>reg_xerr=180+j<em>40+2</em>j;</td>
</tr>
<tr>
<td>reg_yerr=80+i<em>40+i</em>2;</td>
</tr>
<tr>
<td>if (180+j<em>40+2</em>j &lt;= pixel_col + sq_size &amp;&amp; 180+j<em>40+2</em>j + sq_size &gt;= pixel_col &amp;&amp; 80+i<em>40+i</em>2 &lt;= pixel_row + sq_size &amp;&amp; 80+i<em>40+i</em>2 + sq_size &gt;= pixel_row )</td>
</tr>
<tr>
<td>sq_err&lt;= 1;</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>sq_err&lt;= 0;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>.............</td>
</tr>
</tbody>
</table>
//define one droplet sample code
always@(pixel_row, pixel_col)
    begin
        if (sq1_x_pos <= pixel_col + sq_size && sq1_x_pos + sq_size >= pixel_col && sq1_y_pos <= pixel_row + sq_size && sq1_y_pos + sq_size >= pixel_row )
            sq1 <= 1;
        else
            sq1 <= 0;
    end

Table 5.3: Droplet design

5.3.2 BIST Circuit Design

Unfortunately, due to the limitation of DMC device design, the BIST design in real digital microfluidics industry cannot be as ideal as we discussed in Section 5.2. It mainly depends on the tester to load test droplets and clear these droplets after every test pattern finishing its test process. In the mean time, test engineers usually use microscope to observe the output response in current industry, and thus it is hard to automatically send the result back to the comparator. In addition, unlike VLSI memory BIST [32-36], DMC BIST is relatively simple for the controller design; we can even ignore the controller design boundary and combine the circuit of controller directly with the test pattern generator. Thus, in this section we present an applicable BIST design, which is mainly the design of a test pattern generator.
Based on the above discussion, the BIST structure is designed as shown in Fig. 5.5. It consists of two parts. One is the BIST logic circuit, which is designed to generate relative DMC array coordinates for some particular input cells. The other is the VGA logic circuit, which is designed to merge the relative coordinates with some particular (initial) addresses of input cells to complete a test pattern generation and provide interface with the display module.

In fact, the test droplets change positions according to the changing of relative coordinates, while the start cells’ coordinates are fixed.

The purpose of the BIST logic circuit is to implement the fault type distinguishing test algorithms, which include march-d and march-p. Instead of using the Start signal in Fig. 5.2, Two control signals enable 1 and enable 2 are used to active and select the BIST function, and then the BIST start to generate the corresponding test patterns for the DMC. When enable1=enable2=0, the BIST circuit idles and normal operation are performed. When enable=1 and enable2=0, march-d is executed. According to March-d algorithm, reg_x, reg_y, reg_x_5, reg_y_5, reg_x_10 and reg_y_10 are the relative coordinates of the droplet locations.
to the initial or previous droplet locations in three scheduled time with 5 steps difference (Fig4.14). When the BIST logic is selected to implement the march-p algorithm, all the three relative coordinates are active at the same time controlled by both enable signals.

```verilog
always@(posedge clk)
begin
    if(enable1==1&&enable2==0)
    begin
        cnt<=cnt+1;
        if(cnt<43)
            begin
                reg_x<=1;
                reg_y<=0;
            end
        if (cnt>=43)
            begin
                reg_x<=0;
                reg_y<=1;
            end
        if(cnt==84)
            begin
                cnt<=0;
            end
    end
end
```

Table 5.4: Verilog code of generating reg_x and reg_y

The Verilog code of generating relative coordinates reg_x and reg_y are executed at T=0 based on the march-d algorithm shown in Fig. 4.13 is given in Table 5.4. Because in every clock cycle each droplet for march-d only moves one unit pixel and each cell has 40 * 40 pixels with 2 pixels of gap, we let the droplet move 42 cycles to traverse one single cell. A counter cnt is designed to count the number of steps. The way to generate other relative coordinates (e.g.,
reg_x_5, reg_y_10) is similar, except that extra counters are required to implement the scheduling procedure. Except march-d, we also have to achieve partial steps of the march-p algorithm by assigning enable 1 and enable 2 to 01 to implement the entire fault type distinguishing test strategy (Fig. 4.22). The way to generate the relative coordinates based on march-p is very simple just assign the value of x direction and y direction at a constant number.

Based on the above discussion, a fault type distinguishing BIST logic was successfully developed. A synthesized BIST logic gate level schematic design is given in Fig. 5.6. Modelsim [37] functional simulation in Fig. 5.7 verifies the BIST logic design. In Fig. 5.7 (a), enable1 is set to 1 and enable 2 is set to 0. We can clearly see that reg_x_5 and reg_y_5 generate test patterns 5 steps later than reg_x and reg_y; reg_x_10 and reg_y_10 are 5 steps
fall behind reg_x_5 and reg_y_5. It exactly fulfills the functions of the march-d algorithm. In Fig. 5.7 (b), all the cells are active at the same time from west to east. We need to run one time of this step for even row cells and one more time for odd row cells by loading droplets to those cells alternately.

Fig. 5.7 (a): March-d algorithm simulation

Fig. 5.7 (b): March-p algorithm simulation
After finishing the BIST logic design, we must implement the VGA logic to merge the generated relative coordinates with some particular locations of inlet cells to generate a set of complete test patterns. The droplets in the inlet cells following the merged test patterns can reach the target cells. For example, if the inlet cell is (1, 1) and reg_x and reg_y are assigned to 10 respectively, then the droplet will route starting from cell (1, 1) all the way to the end of east. With the changing of relative coordinates, the test paths can be successfully implemented. The Verilog sample code about designing test patterns by changing of relative coordinates is shown in Table 5.5. The key idea is that we define the coordinates of a droplet’s initial position and add the relative coordinates to that initial coordinates to get the next position’s coordinates iteratively.

```
reg [9:0] sq1_x_pos = 180;
reg [9:0] sq1_y_pos = 80;// set initial position’s coordinates
---------------------------------------------------------------------
always@(posedge vsync)
begin
    //FIRST priority DROPLET moving
    if(sq1_x_pos==514||sq1_x_pos==reg_xerr&&sq1_y_pos==reg_yerr)
        begin
            sq1_y_pos<=sq1_y_pos+0;
            sq1_x_pos<=sq1_x_pos+0;
        end
    else
        begin
            sq1_x_pos<=sq1_x_pos+reg_x;// get the next x coordinate
            sq1_y_pos<=sq1_y_pos+reg_y;// get the next y coordinate
        end

```

Table 5.5: Partial Verilog code of merging test patterns
After combining the BIST logic with VGA logic, the design of the BIST circuit is completed. We will verify the function of the entire BIST circuit design in the next section by visualized emulation.

5.3.3 Emulation

Fig. 5.8: RTL diagram of BIST blocks

Fig. 5.9: Emulation result
In order to verify the BIST design, we emulate the generated test patterns through the display logic. Fig. 5.8 shows only the RTL circuit block of the BIST circuit for a DMC diagram. In Fig. 5.8, we see the BIST logic generates signals of relative coordinates for the VGA-logic, and then the completed merged test patterns generated by the VGA logic can be demonstrated through the display module. Snapshots of partial emulation results correspond to Fig. 4.3 and Fig. 4.14 are given in Fig. 5.9.

In summary, the emulation result mimics the chip testing behavior and further verifies the logic correctness of our test strategy developed and the BIST circuit designed. This emulation platform can also be used as a basic DMC-BIST EDA tool, though this EDA tool cannot mimic the real analog behavior of droplet movements.
Chapter 6

Conclusion and Future work

This thesis presented three high level fault models which can cover most of fault behaviors involving two cells and developed a new test strategy for digital microfluidic chips that can distinguish different types of fault behaviors without increasing too much test time. Problems such as flooding and design for testability have been discussed and reasonable solutions are given.

A detailed BIST circuit has been designed to generate test patterns and monitor output responses for digital microfluidic chips, and Modelsim simulations have been carried out to verify the BIST design. A DMC-BIST system emulation software was developed by Verilog and implemented by an ALTERA DE2 FPGA board. The results are very encouraging.

The fault models developed before by other researchers are is not complete especially they miss tooth-missing fault and diagonal coupling fault, because the search domain of physical defects is too large to consider full. Our high level fault models can reflect all
physical defects involving two cells. Compare with [15], our test strategies have detailed algorithms, high fault coverage and can distinguish fault types. Compare with [16] which has test length with complexity $O(n+m)$ and previous test methods [8, 13], our test strategy has smaller test time ($O(n)$) and high fault coverage (100%). Due to their fault model limitation, all test strategies developed before cannot faithfully screen a faulty chip. The test times for all the previous works which did not use parallel scan can be as high as $O(N^2)$.

Unfortunately, for our fault type distinguishing test method, it fails to distinguish FX, PFX and PFXy faults when the testing droplets detect FX or PFX faults in the first step (march-d).

In the future, we will study the following topics to further enhance the test quality, dependability, and CAD tool for digital microfluidic chips.

1. We should figure out the limitation we met in this thesis for fault type distinguishing test method.
2. We should consider more in behavior of droplet moving, such as droplet faulty behavior involving more than 2 cells. In addition, we should develop methods to deal with arrays, such with different structures such as $N \times M$ rectangles or even for T-shape and H-shape.
3. We will develop new detector deployment methodologies such as how to insert each detector into the best locations to further increase the fault coverage, and even increase the on-line test and repair accuracy.
4. We will further investigate the diagnosis and repair methodologies based on the fault
models proposed in this thesis.

5. Try to implement a real tape out BIST circuit, and overcome the random pin assignment problem for large scale chips.

6. Combine the digital and analog functions together to make each droplet movement switched between discrete mode and continuous mode. Based on this technique, we can develop a universal test methodology.

7. With the advance of digital microfluidic chip technology, different application-specific chips will be integrated into a common board. Thus, we need to develop a board-level test methodology like boundary scan test in digital VLSI.
Bibliography


February 2011.


[22] Yuguang Liu, private communication.


