University of Cincinnati

Date: 10/9/2015

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

It is entitled:

FPGA-based fault tolerant design and deterministic routing-based synthesis for Digital Microfluidic Biochips

Student's name: **Onkar Todakar**

This work and its defense approved by:

Committee chair: Wen-Ben Jone, Ph.D.

Committee member: Rashmi Jha, Ph.D.

Committee member: Ian Papautsky, Ph.D.

19551

FPGA-based fault tolerant design and Deterministic routing-based synthesis for Digital Microfluidic Biochips

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 (M.S.)

in the Department of Electrical & Computer Engineering

of the College of Engineering & Applied Science

by

Onkar Vasant Todakar

B.E., University of Pune, 2009

October 9th 2015

Committee Chair:	Dr. Wen-Ben Jone
Committee Members:	Dr. Ian Papautsky
	Dr. Rashmi Jha

Abstract

Microfluidic biochips have been widely used as an alternative to traditional laboratory equipment. They offer a considerable advantage over traditional equipment when the reduction in cost, area and efforts is considered. A lot of research has been done on designing general purpose, cost-effective architectures and also on methods to automate the mapping of assays on to these biochips. Biochips are susceptible to failures due to various reasons such as manufacturing defects, wear and tear etc. We propose a fault tolerant scheduling algorithm which reconfigures the DMFBs in the presence of such faults. A faulty module (for example a mixer with 2×5 electrodes) can be reconfigured using a droplet routing approach that routes droplet, avoiding the faulty electrodes. We observe an average 23% reduction in the assay completion time, when compared to a DMFB with a faulty module. We further extend this routing-based approach to propose an algorithm to map assays to DMFBs. Most of the previous work on mapping assays assumes the presence of virtual modules on DMFBs and schedules operations on them. In our work we propose a deterministic greedy algorithm that routes the droplet on a random sequence of electrodes rather than restricting it to a virtual module to execute the operation. Our algorithm moves the droplets on the DMFB such that the operation is completed in the minimum possible time. The results show approximately 43% reduction in assay completion time, when compared to traditional module based mapping algorithm on a FPGA style DMFB array, and 26% improvement compared to the randomized routing - based synthesis algorithm GRASP.

Acknowledgements

I would like to express my gratitude to my research advisor Dr. Wen-Ben Jone for guiding me at every stage of my research. His constant support, feedback and guidance has been crucial in shaping this thesis. I would also like to thank Dr. Ian Papautsky and Dr. Rashmi Jha for agreeing to be on my thesis committee and for spending time in reviewing my work.

I would also like to thank my parents and my brother for their support and encouragement in every walk of life. Finally, I am thankful to my girlfriend Priti for always standing by my side and for being a constant source of strength and inspiration in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Lab on chip (LOC) refers to devices that miniaturize several laboratory processes like mixing, heating, detection etc. such that they can be included on a small surface like that of an electronic chip. LOC uses micro-electro-mechanical systems (MEMS) and principles in microfluidics to control and manipulate tiny nano-liter droplets on the surface of the chip. Owing to advances in manufacturing, the cost of fabricating these devices is getting lower, which makes them much more economical for widespread use. Advances in miniaturization in chemistry, physics and biology, microfluidics, lithography etc. are pushing LOCs to the forefront of drug discovery, health diagnostics and monitoring. LOCs are used for various applications like personalized medicines, early diagnostics of disease based on biomarkers, nutrition diagnosis etc.

LOCs offer several advantages over traditional laboratory work. LOCs use small volumes of samples and reagents which reduce the cost of the assays as well as the wastage of difficult to obtain samples and expensive reagents. LOCs are faster and have better response time when compared to the laboratory approach. LOCs also substitute the need for large and expensive lab space and lab equipment. Another major advantage of LOCs is automation, which helps reduce intermediate steps as well as the probability of human error.

Commercial applications of LOCs are being introduced in the market. For example, Med-

imate a company in Netherlands has a device to measure concentration of lithium in blood [1]. People with bipolar disorders are treated with lithium and maintaining the concentration of lithium in blood is critical. Medimate's devices help rapid detection of lithium concentration in blood which helps doctors provide on-spot personalized medicine. Medimate also has a device to measure the concentration of sodium in urine. It is very important to keep track of daily sodium intake while being treated for hypertension or cardiovascular diseases. In [2], the author also discusses applications of LOCs like measuring fertility and cancer diagnostics. The widespread of smartphones helps LOCs couple with them to provide a wide range of applications. Researchers have demonstrated various smart phone based LOC applications like colorimetric analysis of serum for cholesterol detection, smartphone microscopy, electrochemistry analysis etc. In [3], the authors develop a RFID-based sweat sensor which measures the proportion of electrolytes in sweat. This can be used in applications such as measuring hydration or the concentration of electrolytes etc. The work in [4] demonstrates a $5 alternative to the expensive cytometers that can be used to determine if a person has HIV and to track its progress. Such applications are extremely helpful in countries with limited healthcare resources. Devices like these aid in fast and cheap diagnostics. According to the forecast in [5] the lab on chip market is expected to grow at a CAGR (compound annual growth rate) of 18%, and will be valued at $13 billion by year 2021.

1.1 Digital microfluidic biochips

There are different types of LOCs based on their construction, actuation mechanisms etc., these types are discussed in detail in Chapter 2. In this thesis we deal with a type of LOCs called Digital Microfluidic Biochips (DMFBs). This is a droplet based biochip consisting of an array of electrodes on which droplets are discretely manipulated. The applications/assays executed on these biochips are made up of series of operations like dispensing, mixing etc. These operations can be executed by moving droplets on a group of electrodes. The major

advantage of DMFBs is that each droplet can be controlled individually which makes the DMFBs general purpose and easily reconfigurable.

DMFBs are complex devices which combine solid state electronics with micro structures to execute microfluidic operations. Since they are used in various critical applications like medical diagnosis, environment monitoring etc., high reliability is a key requirement. DMFBs exhibit unique defect and failure mechanism due to the underlying mixed technologies and multiple energy domains. Various types of faults can be present on a DMFB chip. Some of these faults manifest as a result of defects during manufacturing, while others may be because of degradation due to excessive use. These faults can be classified as catastrophic faults when the defects result in permanent failure of one or multiple electrodes, and as parametric faults when the defects cause variation in parameters like size or concentration of the droplet. Various methods have been proposed to determine the presence of such faults during manufacturing and operation [6].

Several methods have been proposed to reconfigure DMFBs in the presence of faults. For a DMFB array, operations are executed by assuming virtual modules on the surface of the DMFB. In the presence of faults, these virtual modules are placed on the surface of the DMFB such that they do not overlap any faulty electrodes [7]. This reconfiguration technique can be used for both faults detected during manufacturing and operation execution. FPGA-styled DMFB arrays are proposed in [8], [9] to reduce the number of pins needed to control each electrode on the array and to speed up operation execution. These FPGA based arrays have predefined and pre-placed modules. In this work, we propose a reconfiguration method based on droplet routing to make the modules on these arrays fault-tolerant.

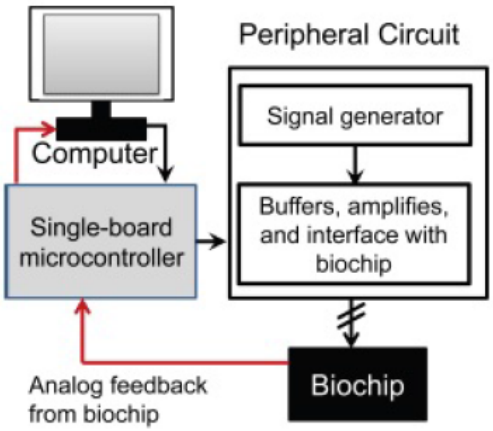

Figure 1.1: DMFB setup [10]

The DMFB chip shown in Fig 1.1 is connected to a microcontroller which controls each electrode on the DMFB individually. To execute an application on the DMFB, the microcontroller stores a series of electrode activation sequences corresponding to the operations constituting the assay. For simple assays, these electrode activation patterns can be generated manually. As the complexity of the applications and the size of the DMFB chip increases, generating the corresponding electrode activation sequences becomes complicated, giving rise to need of design automation. The process of generating electrode activation sequences for an application to be executed on a DMFB array is called *synthesis*.

Various approaches have been proposed in the literature to solve the synthesis problem. The most common approach is assumption of virtual modules to execute the operations like mixing, splitting etc. With virtual modules, all electrodes that are part of module are assumed to be occupied during operation execution. The virtual modules also require a segregation layer of one electrode width around the module to prevent accidental mixing of droplets. This approach blocks a large number of electrodes for a single operation, thereby reducing the number of operations that can be executed in parallel. As an improvement to this approach, Maftei et.al [11] proposed a droplet aware execution mechanism to get rid of the segregation layer around the module. Since operations can be executed by routing droplets on any sequence of electrodes, they propose a droplet routing based methodology to

solve the synthesis problem. In this work we improve the routing based algorithm proposed by Maftei et.al by making the algorithm deterministic and greedy.

1.2 Contributions

- We determine the type of faults encountered on the modules in the FPGA-based general pupose field programmable (GPFP) architecture proposed in [8]. Depending on the type and location of faults, we reconfigure these faulty modules and determine their operation completion time using droplet routing based operations. We implement a fault tolerant scheduling algorithm which uses these reconfigured modules rather than simply discarding them. We observe an average 23% improvement in assay completion time by using reconfigured faulty modules instead of discarding them.

- We improve upon the routing based synthesis methodology proposed by [11]. The approach in [11] uses a randomized approach to determine the direction for a droplet to move during operation execution. We propose a deterministic greedy algorithm that selects the best possible direction for the droplet to move at each step. Our algorithm improves the assay completion time by 30% on an average when compared to the routing based synthesis algorithm proposed by Elena et al.

1.3 Thesis organization

- Chapter 2 includes some background information to get a better understanding of the microfluidic biochips. We discuss various types of microfluidic devices and their characteristics. We define the architecture of DMFBs, the working principle and various electrode addressing schemes. We review various synthesis approaches proposed in literature. We also discuss about faults and their effects on DMFBs. We review the

GPFP architecture and various faults it is susceptible to.

- Chapter 3 first discusses the completion time for routing-based operations. We characterize the routing based operation completion, and use the results to calculate the mixing time of reconfigured modules on the GPFP architecture. We then discuss the fault-tolerant list scheduling algorithm and evaluate it.

- Chapter 4 extends the routing-based operation completion methodology to synthesis. We propose routing based synthesis (RBS), a deterministic and greedy routing based algorithm which is an improvement over GRASP proposed in [11].

- Chapter 5 discusses conclusion and future work.

Chapter 2

Background

In this chapter we provide some background information for better understanding of the field of LOCs. We begin the chapter with a brief description of different types of LOCs. We describe the construction and working principle of DMFBs. We then briefly cover other topics like synthesis, faults and fault tolerance.

2.1 Working of LOCs

LOCs implement various applications like detection of glucose in blood,urine etc. by moving, mixing, separating and observing small amounts of liquids on their surface using principles in microfluidics. Microfluidics is an interdisciplinary field that applies various concepts from physics, chemistry, biology, engineering etc. to design systems which manipulate extremely small volumes of liquids. These devices can be classified into two main categories based on the mechanisms they use to actuate the liquids. The first category is called passive microfluidics which employs passive techniques like capillary forces [12] for liquid movement. The second category is active microfluidic devices which force the droplet movement using internal or external devices like micropumps, microvalves [13] or using electrokinectic mechanisms [14]. In this thesis, we only consider active microfluidic devices.

Active microfluidic devices are further categorized based on their architecture. Channel-

based devices have etched micro channels through which samples and reagents flow either continuously or as discrete droplets using either internal or external forces. Using droplets over continuous flow of liquids has advantages like better control over liquid volume, better mixing operation and high throughput [15]. Various flow control mechanisms like focused flow [16], T-shape generators [17] etc. are used to generate discrete droplets in a channel based microfluidic device. These devices are cheap and are easier to manufacture. Since each fluid has dedicated channels, there is very little probability of droplet contamination. On the other hand, since these devices are manufactured with pre-etched channels they are application specifc and hence have limited re-configurability. To overcome this limitation, a completely reconfigurable, general purpose device called digital microfluidic biochip (DMFB) is proposed.

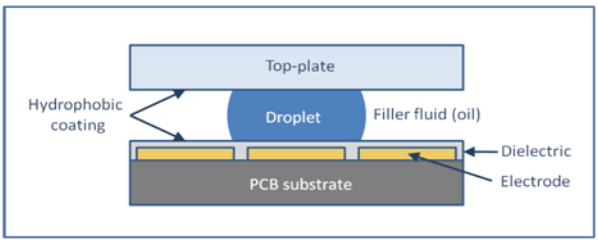

Figure 2.1: Cross section of a DMFB array

A DMFB is an array of special electrodes on which discrete nano-liter sized droplets are manipulated using a phenomenon called *electrowetting on dielectric* (EWOD). The DMFB array is made up of two parallel glass plates of indium tin oxide, seperated by a tiny gap between which the droplets are sandwiched. The bottom plate has a patterned array of individually controllable electrodes and the top plate is continuous ground [18]. Both top and bottom electrodes are coated with a dielectric to reduce the wettability of the surface. The construction of a DMFB array is explained in detail in [19]. The DMFB also has reservoirs for samples and reagents which are connected to the input ports of the chip. It also has output ports to collect waste droplets. In addition to ports, the DMFB can also have other peripherals like heaters, optical detectors etc. EWOD is defined as the modification

of interfacial tension between droplet and the electrode due to electric field applied at the surface of the electrode. If the voltage is applied only on one side of the droplet, as shown in Fig 2.1 a gradient in surface tension is created which causes the droplet to move in the direction of the activated electrode.

The sample droplet is enclosed in a filler medium like silicone oil, and is sandwiched between the electrodes. The droplet is sized such that it overlaps with the adjacent electrodes. To move a droplet, we deactivate the electrode it is currently on and activate its neighbor. EWOD causes gradient in the interfacial tension between the droplet and the electrode, thereby making it move towards the activated electrode. Each assay is made up of a series of dispensing, transport, mixing, splitting and detection operations. Each operation executed on a digital microfluidic biochip can be broken down into a series of simple operations which can be performed by repeatedly routing the droplet on a series of electrodes. Droplet transport in a particular direction is implemented by activating an electrode adjacent to the droplet in that direction. Dispense operation can be considered to be made up of several individual transport steps. To merge two droplets, they are brought closer such that they are separated by one electrode. We then activate the middle electrode such that both droplets move towards it, effectively merging both droplets. To split a droplet, both electrodes on either side of the droplet either in X direction or Y direction are activated such that the droplet tries to move in both directions causing it to split. These operations are depicted graphically in Fig 2.2.

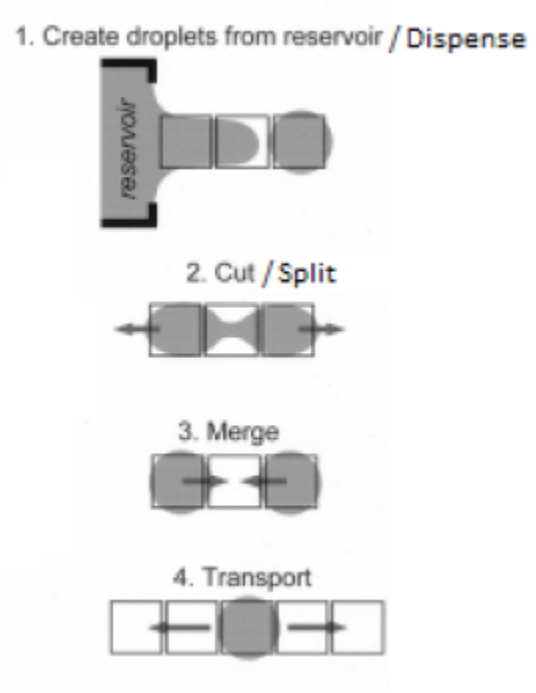

Figure 2.2: Operations on a DMFB array

2.2 Electrode addressing on DMFBs

To move each droplet on the DMFB array discretely, each electrode needs to be controllable individually. The process of assigning these individually controllable electrodes to external pins to send control signals to them is called *electrode addressing*. Early DMFBs used direct addressing in which each electrode was assigned to an individual pin as shown in Fig 2.3. Since each electrode can be directly controlled from external inputs, this offers great flexibility. Direct addressing is extremely beneficial for smaller DMFBs; but as the size increases, so does the complexity. Routing wires on the substrate under the DMFB electrodes gets complicated and expensive to manufacture. This also makes the DMFB more susceptible to faults.

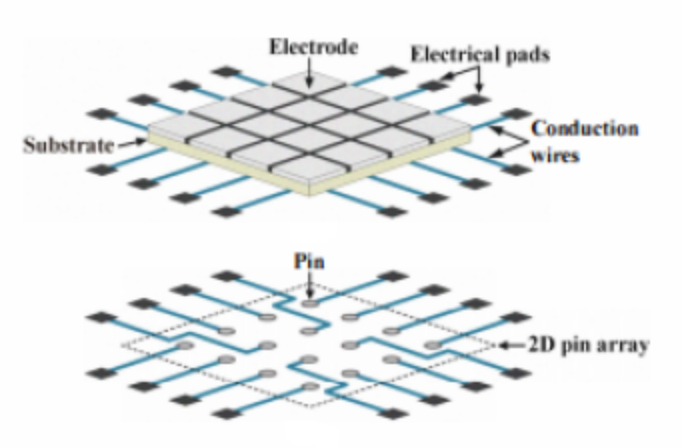

Figure 2.3: Direct addressing [8]

To reduce the pin-count, various pin constrained methodologies are proposed which assign a single pin to a group of electrodes. in [20] the authors propose a pin constrained design using a multi-phase bus in which every n^{th} electrode is connected to a single pin, thereby reducing the pin count to n. In [21] Hwang et al. propose an array partioning scheme to determine pin assigment such that each set of pins correspond to a partition. This method is improved in [22] by making the partioning droplet aware. Xu and Chakrabarty in [23] proposed a broadcast addressing scheme by which pin count is reduced by grouping electrodes together. They first synthesize a given assay assuming a direct addressed DMFB to obtain the electrode actuation sequences. These sequences are then partitioned into groups such that each group is controllable individually via a single pin. This approach uses the direct addressing based routing information as its input to reach broadcast addressing based routing. Using the broadcast addressing scheme also makes the array assay specific. Thus this approach cannot be used for general purpose field programmable applications.

To overcome the limitations of direct addressing, [24] proposed a cross referencing approach that uses M+N electrodes to control droplets on an M×N chip. In this method, electrodes are in the form of orthogonal rows. To move a droplet from on location to other, we activate the row and column corresponding to the new position. In [24] the authors proposed a modified cross referencing approach that allowed multiple droplets to be con-

trolled simultaneously. There is an inherent limitation: the number of droplets the device can simultaneously control due to the possible electrode interference.

A novel architecture called field programmable pin constrained (FPPC) is proposed by Grissom et al. in [9] which retains the benefits of pin constrained methodologies and is field programmable such that it can be used for any application. The topology of the FPPC architecture is shown in Fig 2.4. On DMFB architectures which are in the form of electrode arrays, reconfigurable operations like mixing, splitting etc. can be implemented on any electrode. The FPPC architecture, on the other hand, has dedicated regions/modules to implement specific applications. The numbers on the electrodes specify the group to which they belong. All electrodes belonging to any particular group are connected to a single external pin. On the left side, we see four 2×4 modules which can be used for operations like mixing and merging. All four modules use identical group of electrodes (electrodes 7 to 13) as shown in Fig 2.4. Using these modules we can mix 4 droplets simultaneously by activating those electrodes. In addition to the mixing modules, on the right side, we have 6 modules that can be used for split, store and detect operations. The FPPC uses multi-phase bus proposed in [20] for routing. Electrodes labelled 1-6 are used for routing.

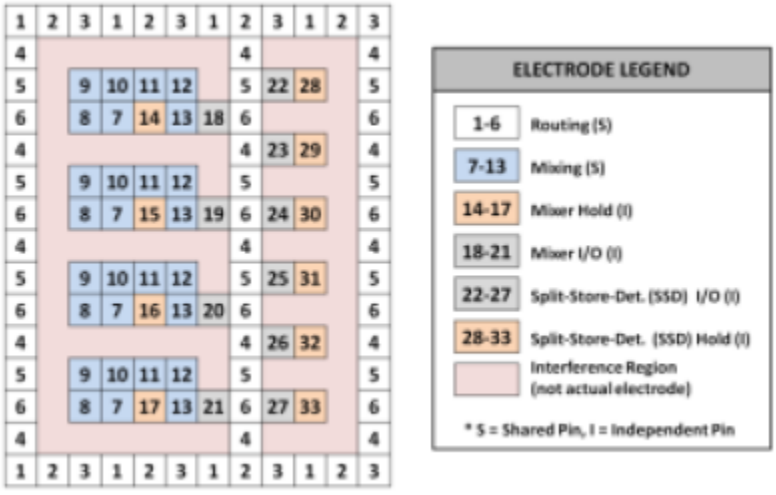

Figure 2.4: Architecture of FPPC DMFB [9]

The synthesis process on the FPPC architecture is simpler, because the allocation and

placement steps are already taken care of. Routing multiple droplets over pin constrained electrodes is complicated and computationally intensive due to electrode interference. Even though FPPC is a general-purpose architecture that can implement any assay, the performance of the FPPC array method is worse than direct addressing arrays. In [8] Rissen et al. proposed an improved version of FPPC called general purpose field programable (GPFP) architecture whose performance matches direct addressing arrays. The details of the GPFP architecture are described later in this chapter.

Active matrix (AM) addressing is another electrode addressing methodology that retains the benefits of direct addressing arrays. Using AM addressing, an M $\times$ N array can be controlled using M + N pins. To select an element at (m, n), pulses of appropriate pulse width are applied to the row and column select lines. In [25], the authors implement an AM addressing system using thin-film transistors (TFTs) and integrate them with EWOD electrodes. This addressing scheme requires special fabrication techniques to help electrodes hold charge during off pulses.

2.3 Overview of GPFP DMFB architecture

The GPFP DMFB architecture proposed in [8] shows significant improvements over the FPPC architecture proposed by Grissom and Brisk in [9]. The architecture proposed in [8] is the first to use scan chain-based electrode control mechanism. The scan chain is made up of scan and hold flip-flops, one for each group of labelled electrodes. This drastically reduces the number of external pins required to control the array. The number of pins required by the GPFP architecture is equal to twice the number of scan chains on the array. The routing approach on GPFP is similar to that on the FPPC. They differ in the arrangement and structure of modules. The GPFP architecture has eight 2$\times$5 electrode modules which can be used for any reconfigurable operation. Each module has a dedicated electrode (orange electrode) that can be used for operations like storage and detection. Another difference is

that each module has its own group of pin-constrained electrodes which allows all modules to work independently. The assay execution time is 40% faster compared to the FPPC architecture of similar size. In spite of these advantages, there are a few drawbacks of the GPFP architecture. They are discussed below.

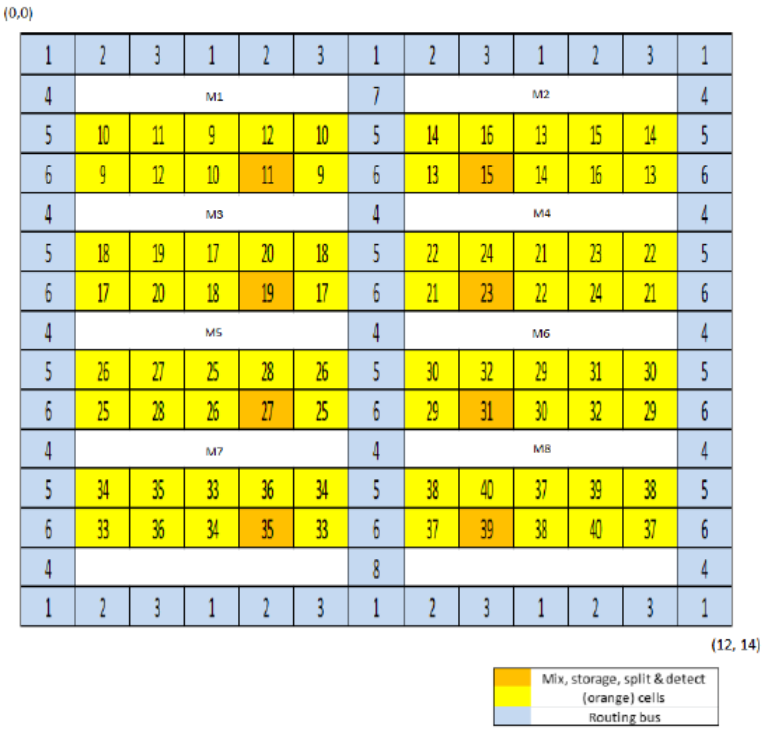

Figure 2.5: Architecture of GPFP DMFB [8]

Electrode utilization is low

The total number of electrodes in this 13×15 GPFP architecture in Fig 2.5 is 145. Out of these 145 electrodes, 65 electrodes (colored in blue) are dedicated to routing, i.e., approximately 45% of the electrodes are idle during operation execution. This will cause non-uniform wear in the electrodes used for routing versus the electrodes used for reconfigurable operations. Reconfiguring these electrodes such that they are utilized during operation execution will help speed up the execution time.

Routing overhead not considered

The results mentioned in [8] compare the GPFP with the FPPC architecture. The FPPC

architecture calculates the routing overhead introduced by its pin-constrained nature, but GPFP on the other hand does not mention the routing overhead involved. Considering limited routing resources and its pin constrained architecture, the routing overhead introduced by GPFP should be significant. Routing droplets between modules has to be sequential. Fluidic constraints between droplets need to be maintained. The GPFP architecture has much higher routing complexity than a simple 2-dimensional array of electrodes.

Difficult to scale

For larger assays we need more resources to speed up their execution. To increase the number of modules, we need to extend the existing architecture. There are two possible ways to extend the GPFP architecture. The first option is to add more modules under the existing modules. In this scenario, the number of rows available for routing remains the same, thereby increasing the routing complexity even more. The second option is to add another column for modules next to the existing column of modules. Even in this scenario transporting droplets between columns would be complex. Thus its difficult to scale-up this architecture without any modification.

Highly susceptible to faults

In addition to the catastrophic and parametric faults discussed later in this chapter, GPFP is also susceptible to failures in the underlying scan-chain based actuation circuitry. Different faults impacting GPFP are discussed later in this chapter.

2.4 Synthesis

DMFBs execute the required operations on droplets by actuating electrodes in a particular sequence such that they are moved, mixed, etc. For simple assays it is possible to schedule operations and generate electrode patterns manually. But for applications like drug discovery, where we iterate over various combinations and concentrations of samples and reagents, this manual translation is too repetitive and inefficient. Also in the case of large assays,

the manual translation may not lead to the best possible solution. Hence, we use design automation techniques from the EDA (electronic design automation) industry to automate this translation. Considering the similarities between digital microfluidic biochips and digital electronics, we call this translation process as *synthesis*. Synthesis can be defined as the process of mapping an assay on to any given DMFB array and generating the electrode activation sequences.

The biochemical application or assay that we want to execute on the DMFB array is modelled as a directed acyclic graph. Each node in the graph represents a operation performed in the assay and each edge determines the dependencies between those operations. Alternatives to graphs are programming languages like BioCoder [26] that describe applications in a form much easier for automation tools. Either one of these approaches can work well with the synthesis tools. We also have a module library which contains information about the dimensions and execution time of various modules like mixers, dispensers, detectors etc. We also model the DMFB array in terms of its dimensions, input output ports and peripherals.

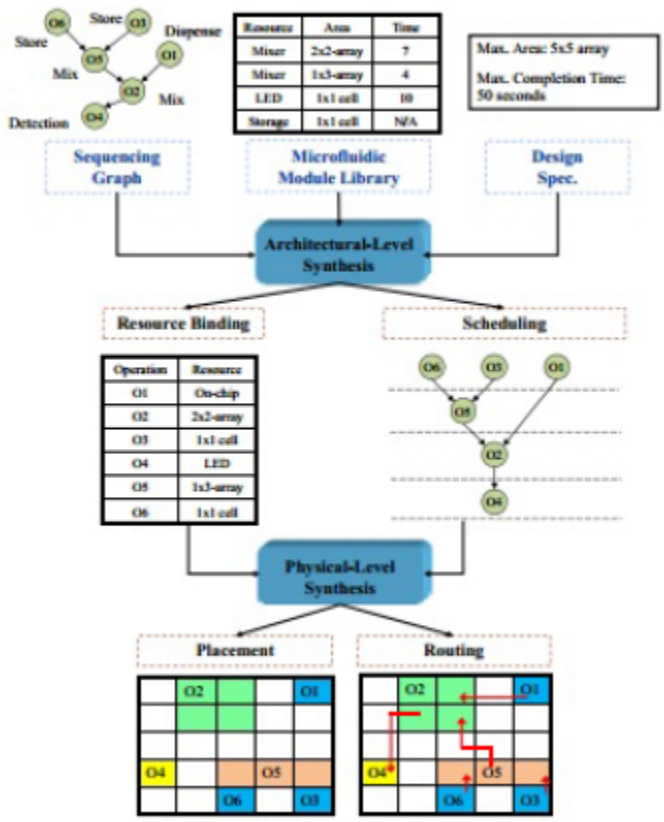

Figure 2.6: Steps involved in Synthesis [8]

The synthesis problem can be broken down into two parts namely *architecture level synthesis* and *physical level synthesis*. In architecture level synthesis we schedule all operations such that all dependencies between them and resource constraints are met. These operations are scheduled on the selected modules and with a simplified estimate of the placement. The goal of this step is to generate the best solution such that it minimizes the operation completion time. This is followed by physical synthesis, which deals with placement of the modules based on the previous step and routing of droplets between them. The scheduling step of architectural synthesis has proven to be NP-complete. In [27], the authors propose ILP and heuristics algorithms. i.e., List scheduling algorithm and genetic algorithm for architectural level synthesis. They use simulated annealing based method to do the placement of the modules during the physical synthesis step. Even though the two step approach simplifies the problem, the scheduling step works by assuming placement of modules. This may not

be correct at times and the scheduling step needs to be re-run to solve this problem.

The unified synthesis methodology has been proposed by Su et al. in [7] which determines the placement of modules using simulated annealing during scheduling. These results are further improved by Yuh et al. [28] using T-tree data structure. A routing aware synthesis methodology is proposed by Xu et al. [29] that considers routing during scheduling and placement of modules. In [11], the author proposes a unified synthesis methodology based on the tabu search metaheuristic to determine allocation and binding. This current allocation and binding is then used by List scheduling to determine the schedule.

In the synthesis algorithms discussed so far, the reconfigurable modules e.g., mixers are assumed to be fixed during operation execution. Since operations can be executed by routing droplets on any sequence of electrodes, [11] proposes a synthesis approach in which reconfigurable modules are moved during operation execution to avoid space fragmentation, leading to better placement. The approach is further extended by using non-rectangular modules to better utilize the space on the DMFB array. Better placement improves parallelism, thereby allowing more operations to execute in the same time step. All these solutions assume an isolation ring of 1 electrode width around each module to prevent accidental merging with other droplets. [11] further proposes a droplet aware execution approach that prevents accidental merging and this allows placement of modules without the isolation ring, thereby helping better utilization of the chip area. This approach also helps routing by allowing droplets to move over modules.

Since we execute operations on DMFBs by routing droplets on a series of electrodes, these operations can also be executed by routing them on any random sequence of electrodes. The next step in improving synthesis would be getting rid of the virtual modules used for executing operations. In [11] the author proposes a routing-based synthesis method in which each operation is implemented by moving the droplet on a random sequence of electrodes. The direction of the droplet is selected randomly from the list of best possible moves at that time step. This approach shows an improvement of approximately 47% over the module-

based approach.

2.5 Faults in DMFBs

DMFBs are affected by different types of faults which may lead to failures in operation execution or incorrect results. These faults are discussed extensively in [6]. There are 2 types of faults encountered in DMFBs: permanent/catastrophic faults and transient faults.

Permanent faults

Permanent faults are introduced either by manufacturing defects or by aging. These faults lead to failures in operation execution or completion. Here is a list of commonly encountered permanent faults.

- *Dielectric Breakdown* is caused due to high voltages applied to the electrodes. This causes a short between the droplet and the electrode, thereby causing the droplet to be stuck at that particular electrode.
- *Short between neighboring electrodes* causes neighboring electrodes to behave as one big electrode. The droplets on such a large electrode have no overlap with their adjacent electrodes. This makes it impossible for the droplet to move.
- *Breaking down of the insulator* is the result of repeated use or aging. This causes the droplet to break into fragments due to irregularities in surface tension.
- *Open in the electrode control circuitry* results in failure to actuate the electrode, thereby rendering it useless during operation execution.

Transient Faults

Transient faults are intermittent faults that occur during operation execution. These faults do not block the execution of the assay. These may cause incorrect results due to various reasons such as variation in droplet volume or concentration. The most common causes of transient faults are discussed below.

- *Variation in geometrical parameters* such as thickness of insulator, dimensions of the electrode, spacing between the electrodes may result in transient faults.

- *Incorrect overlap between the droplet and electrode* results in droplets of unequal volume during operations like dispense or splitting. This effect cascades over time and results in incorrect operations.

- *Change in viscosity of droplets or filler medium* may be caused due to variation in temperature. This causes incorrect concentration of the fluids which may lead to incorrect results.

- *Cross contamination* may occur due to droplets which have tendencies to get adsorbed on the surface of the electrodes. When other droplets move on to these contaminated electrodes, the purity of the droplets may be compromised. This will result in incorrect results.

2.6 DMFB Testing

As DMFBs are used in various safety critical applications, reliability becomes the key performance parameter. DMFBs are tested for permanent/catastrophic faults and parametric faults by controlling and tracking the movement of droplets on the surface of the microfluidic array. Testing for manufacturing defects such as shorts between electrodes, variation in geometrical parameters should be performed immediately after manufacturing. Other faults like degradation of the insulator, breakdown of the dielectric may occur during operation execution. Various methods for online testing (testing concurrently during operation execution) are proposed in literature [6].

Catastrophic fault testing

Most methodologies proposed for testing catastrophic faults route test droplets across the microfluidic array to locate the faults. A simple testing scheme called parallel scan is

proposed in [6], where droplets are routed row-wise and column-wise on the microfluidic array. The presence of the droplet at an electrode can be determined using a capacitive sensor. In case of catastrophic faults the test droplet would be stuck at the faulty electrode; its absence at the destination electrode can be detected using the capacitive sensor. By row-wise and column-wise routing of the droplet, the exact location of a faulty electrode can be determined. Various sophisticated techniques to reduce test-time, number of reservoirs, detect multiple faults in parallel and concurrent testing etc. are proposed in [6].

Parametric fault testing

DMFBs can have integrated sensors that can facilitate real-time fault detection and correction. A combination of LED and photodiode can be used to measure the concentration of a droplet [30] [20]. A capacitive sensor can be used to determine the volume of the droplet. A CCD camera based system can also be used to measure the volume of a droplet. This CCD based system is complex but is easier to use, since it eliminates the need to route the droplet to specific locations for detection.

2.7 Fault tolerance in DMFBs

The use of DMFBs in critical applications like clinical diagnosis, pathogen monitoring etc. requires a high level of accuracy and reliability. We can improve the reliability and accuracy of DMFBs by making them resistant to both catastrophic and parametric faults. One of the ways to make a system fault tolerant is to add some level of redundancy. For DMFBs we can add redundancies at the device level to deal with catastrophic faults or at the assay level to deal with parametric faults.

To handle catastrophic faults the most straightforward approach is to reconfigure modules such that the faulty electrode is discarded. Su et.al [31] proposed a simulated annealing based placement methodology which allows a module to be placed elsewhere, if the module overlaps with a faulty electrode. A tile-based architecture is proposed in [32] which handles faulty

electrodes with partial or global reconfiguration depending on the operation schedule and resource availability. In [33] parametric faults are handled by considering both time and space redundancies.

2.8 Faults possible in GPFP DMFB

GPFP is susceptible to the catastrophic and parametric faults discussed in the previous section. In this thesis work, we only deal with catastrophic faults that cause a single electrode failure. Various possible causes for single electrode failure are discussed in this section.

Short or Open in the actuation circuitry

The GPFP uses scan chained based architecture to control the electrodes on the chip. A special type of sequential circuit called scan and hold flip-flop shown in Fig 2.7 is used to control the electrodes. The scan and hold flip-flop adds a hold latch at the output of the standard flip-flop circuit. A short or open in the section labelled Flip-flop circuit will result in scan chain failures, thereby causing all electrodes connected to this scan chain to malfunction. For example, with this type of scan chain failures one may have multiple modules M1, M2, M3 etc. shown in Fig 2.5 fail. Similarly a short or open in the section labelled as Hold Latch will cause electrodes connected to that particular Hold latch to malfunction. For example, consider module M1 in Fig 2.5, a failure in the hold latch may cause all electrodes labelled 9, 10, 11 etc. to fail. These type of failures leads to either complete failure of few modules or failure of two or more electrodes in a particular module. In both cases, the entire module is faulty and unsuitable for the fault tolerance method proposed in this thesis.

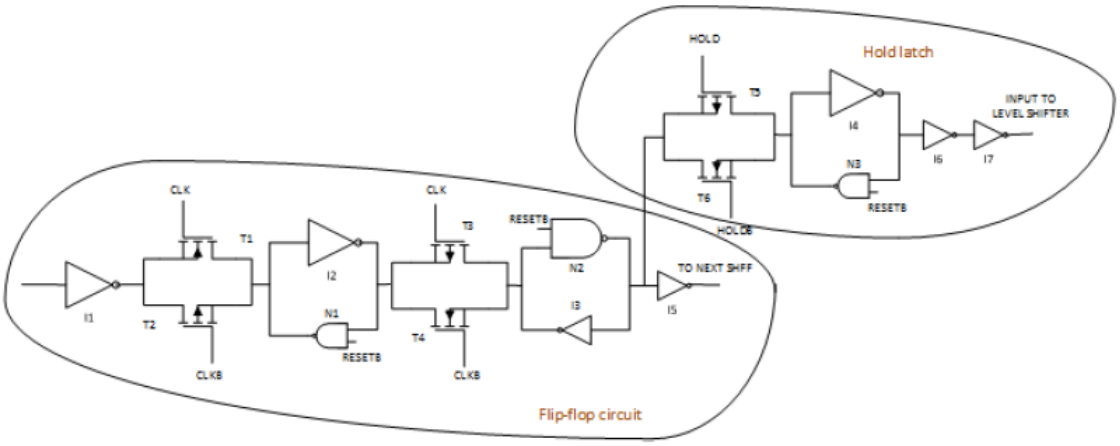

Figure 2.7: Scan and hold flip-flop in GPFP DMFB [8]

Open or Short in the Level Shifter circuitry

The DMFB electrodes need high voltage, typically around 20V for operation. The input from the scan and hold flip-flop is typically around 5V and hence we need a high voltage level shifter for each electrode. The output of each scan and hold flip-flop fans out to the corresponding level shifters connected to the individual electrodes (For each of the electrodes labelled 9 has its own level shifter circuit). Faults in the level shifter circuitry will result in failure in the electrode connected to the faulty level shifter. This type of single electrode failure is considered as a target fault addressed by the fault tolerant algorithm proposed in this thesis.

Permanent Faults

The typical permanent faults like dielectric breakdown, insulator breakdown, and open in control circuitry occur in the GPFP architecture as well. These faults also result in single electrode failure and are considered as target faults in our fault tolerant algorithm.

Chapter 3

Fault Tolerance in FPGA based DMFBs

In this chapter, we develop a fault tolerant approach to deal with catastrophic faults in the GPFP architecture. We start by characterizing operation completion percentage based on the droplet movement. Based on the characterization results, we calculate the mixing time of 2×5 modules in the GPFP architecture reconfigured for different fault types. We then propose a fault tolerant synthesis algorithm for catastrophic faults in the GPFP architecture.

3.1 Characterization of operations based on routing

Paik et al. [34] determine the operation completion time for mixers of various dimensions. The results of their experiment are summarized in Table 3.1. These results are for mixers that have the electrode pitch, i.e., the size of the electrode as 1.5mm and the gap between two electrodes is 600µm. We assume same dimensions for the GPFP architecture used in this experiment.

Table 3.1: Module Library

DIMENSIONS	MIXING TIME (in seconds)
1 × 4 linear mixer	4.6
2 × 2	9.95
2 × 3	6.1
2 × 4	2.9

In [11], Maftei proposes an analytical approach to determine the percentage completion of mixing operation based on its droplets movement. The percentage of operation completion is dependent on the direction in which the droplet moves. A droplet at any electrode (excluding the boundary electrodes and fluidic constraints) at any instant can be moved in one of the five possible directions as shown in Fig 3.1

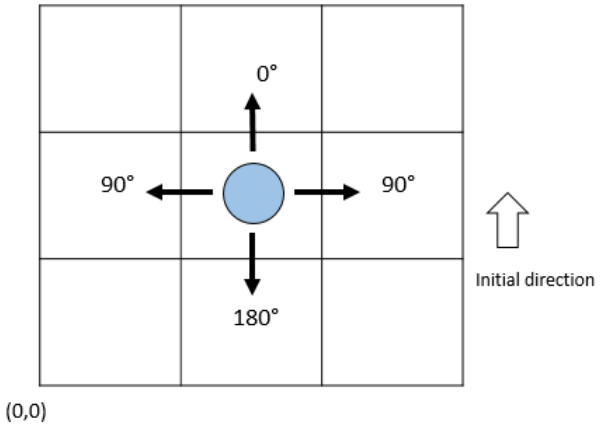

Figure 3.1: Possible routes for any droplet

Consider a droplet *d* shown in Fig 3.1, at time *t*, at location (1, 1). Assume that the droplet moves up north from its initial position (1, 0) at time *t-1*. At time *t+1*, the droplet can move in any of the four directions indicated by the arrows or it can stay at its current location. At time *t+1*,

- If the droplet moves up *north*, we say that the droplet moved in direction 0° with

respect to its current direction.

- If the droplet moves either *east* or *west*, the direction of the droplet is considered to be 90° with respect to its current direction.
- If the droplet moves down *south*, the direction of the droplet is considered to be 180° with respect to its current direction.
- The droplet can also stay at its *current location*. This has no impact on the operation completion time and therefore has no special significance.

We use an analytical approach proposed in [11] to determine the completion percentage of mixing based on its droplet movement. We start by decomposing the movement inside a 2×2 mixer with mixing time 9.95 seconds according to Table 3.1.

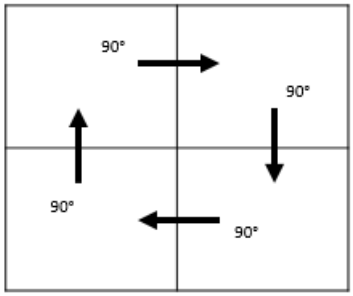

Figure 3.2: Droplet movement in 2 × 2 module

The mixing/rotation of a droplet in a 2 × 2 mixer can be decomposed as 4 steps of 90° movements. Based on the data in [11] we assume that a droplet takes 0.01 seconds per step, i.e., to move from one electrode to the other. Considering the mixing time of 9.95 seconds, we can say that it takes 995 time steps to complete the mixing operation.

If 995 steps = 100 % mixing

1 rotation, i.e., 4 steps of 90° operations = 0.40% mixing

Therefore 1 step of 90 = 0.1 % mixing

Let us consider a 2×3 mixer whose mixing time is 6.1 seconds according to Table 3.1. The movement of a droplet in a 2×3 mixer can be broken down into 6 steps, 4 steps of 90°

and 2 steps of 0° movements. The entire mixing operation is completed in 610 steps.

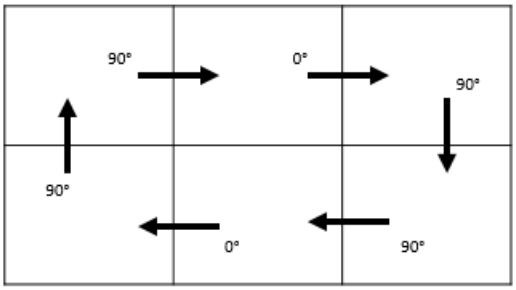

Figure 3.3: Droplet movement in 2 × 3 module

Assuming that 610 steps = 100 % mixing

6 steps (4 steps of 90° + 2 steps of 0°) = 0.98 % of mixing

2 steps of 0° = 0.98 −0.4 (i.e., 4 steps of 90°)

Therefore, 1 step of 0° = 0.29% mixing

The forward mixing percentage (in 0°) is not a constant value. It depends on the number of electrodes the droplet moves in a linear direction, i.e., 0°. In a 20×4 module, the droplet moves 2 electrodes in linear direction. We use the nomenclature 1 - 0° for a droplet which moves one electrode in 0° and 2 - 0° for the droplet which moves two or more electrodes in 0°. The mixing percentage for 2 - 0° can be calculated by decomposing a 2 × 4 mixer. The mixing time of a 2×4 mixer according to Table 3.1 is 2.9 seconds.

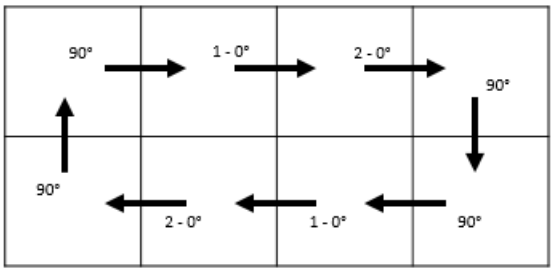

Figure 3.4: Droplet movement in 2 × 4 module

The rotation inside a 2×4 mixer can be decomposed as 4 steps of 90°, 2 steps of 1 - 0° and 2 steps of 2 - 0° movements. In [11], the author assumes that all the 0° movements in a 2×4 module are similar and uses them to calculate a new mixing percentage. The results

in [11] however assign two different values to 0° steps to movements which are similar i.e., the 1 - 0° step in 2×3 mixer and 2×4 mixer respectively. We assume that the step 1 - 0° in a 2×4 mixer is the same as the one in a 2×3 mixer. We consider the mixing percentage calculated in the previous step to calculate 2 - 0 °. The mixing time of a 2×4 mixer is 2.9 seconds which implies 290 steps. Thus we have the following derivations.

290 steps = 100 % mixing

8 steps (4 steps 90° + 2 steps 1 - 0° + 2 steps 2 - 0°) = 2.76 %

2 steps 2 - 0 ° = 2.76 −0.4 −0.58

Therefore, 1 step 2 - 0 ° = 0.89 % mixing

As stated earlier, the mixing percentage increases as the droplet continues to move in its existing direction. In this thesis however, we take a pessimistic approach and consider only 2 values for mixing. For droplets moving 2 or more electrodes in 0°, we consider that the mixing percentage equivalent to 2 - 0° step.

To calculate the mixing percentage for 180° step, we decompose a 1×4 mixer. Its mixing time according to Table 3.1 is 4.6 seconds. The rotation in a 1×4 mixer can be broken down into 2 steps of 1 - 0°, 2 steps of 2 - 0° and 2 steps of 180°. It takes 460 steps to finish mixing on a 1×4 mixer. The value of 1 step of 180° mixing can be derived as follows.

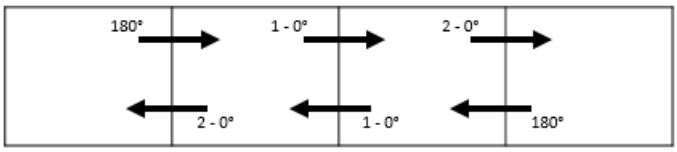

Figure 3.5: Droplet movement in 1×4 module

460 steps = 100% mixing

6 steps (2 steps 1 - 0° + 2 steps 2 - 0° + 2 steps 180°) = 1.304% mixing

2 steps 180° = 1.304 −0.58 −1.78

Therefore 1 step 180° = −0.52 % mixing.

According to [11], negative mixing can be attributed to the unfolding of patterns inside the droplet i.e. droplets have a tendency to separate when moved backwards. The results of

the module characterization experiment are summarized in Table 3.2.

Table 3.2: Mixing percentage based on droplet direction

DIRECTION	**MIXING % COMPLETE**
1 - 0 °	0.29%
2 - 0°	0.89%
90°	0.1%
180°	−0.52%

Let us calculate the mixing time of a 2×10 module based on the results summarized in Table 3.2.

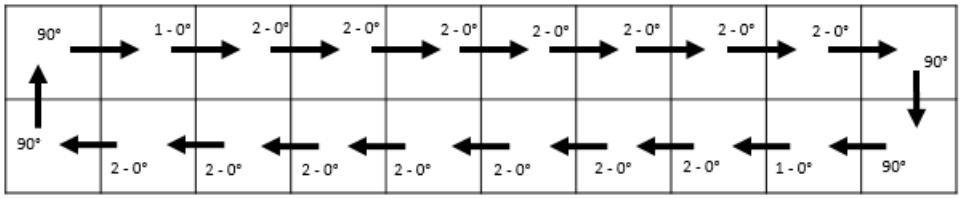

Figure 3.6: Droplet route in a 2×10 module

The route of a droplet inside a 2 × 10 module can be broken down into following steps, four 90° steps, two 1 - 0° steps and fourteen 2 - 0° steps.

$$\begin{aligned}\text{Thus, we have mixing completed in 1 rotation} &= 4\times\ (0.1) + 2\ \times\ (0.29) + 14\ \times(0.89)\\ &= 0.4 + 0.58 + 12.46\\ &= 13.44\%\ \text{mixing per rotation}\end{aligned}$$

The number of rotations needed to complete the mixing equals 100/13.44 = 7.44 rotations. Thus, the time taken for each rotation is 0.2 seconds. Finally, we have the mixing time of a 2×10 module as 7.44×0.2 = 1.48 seconds.

3.2 Module reconfiguration in presence of faults

3.2.1 Single Electrode Fault

In this section we will calculate the mixing time of a reconfigured module in the GPFP architecture with a single electrode fault. The mixing time depends on the location of the fault in the module. Let us assume Module 1 in Fig 2.4 as the faulty module. In this case one of the 10 electrodes is faulty, and the remaining 9 electrodes can be used to route the droplet.

Type 1: Fault in one of the corner electrodes

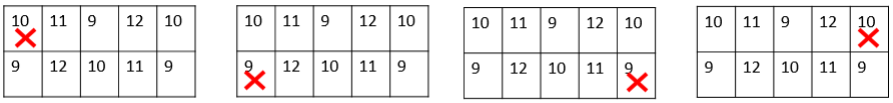

Figure 3.7: Faults in one of corner electrodes

For example, in the presence of a single electrode fault at the corner electrode labelled 10, the droplet can be routed as shown in Fig 3.8.Owing to the symmetry in the module, the path taken by the droplet will be similar to the other Type 1 faults shown in Fig 3.7.

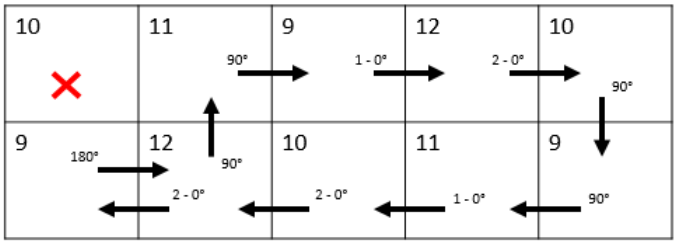

Figure 3.8: Mixing route for *Type 1* fault

Each rotation can be decomposed into steps as shown in Fig 3.8. The routing path is made up of 10 steps, 2 steps in 1 - 0°, 3 steps in 2 - 0°, 4 steps in 90° and 1 step in 180°. The mixing percentage completed in 1 rotation can be calculated as follows.

$$\begin{aligned}\text{Percentage completed in 1 rotation} &= 1\times\ (\text{-}0.52) + 4 \times (0.1) + 2 \times (0.29) + 3 \times (0.89) \\ &= -0.52 + 0.4 + 0.58 + 2.67 \\ &= 3.13\%\ \text{mixing per rotation}\end{aligned}$$

The total number of rotations needed to complete the operation is 100/3.13 = 31.95. Since the time needed for each step (movement of the droplet from 1 electrode to the next) is 0.01 seconds and the number of steps taken to complete each rotation is 10, the time taken for each rotation is 0.1 seconds. Thus the total mixing time is equal to 3.195 seconds. According to Table 3.1, the mixing time of a 2×4 mixer is 2.9 seconds. This mixing time is smaller than the mixing time for single electrode fault shown in Fig 3.8. Thus, if we reconfigure the faulty 2×5 module as a 2×4 mixer rather than as shown in Fig3.8, we can reuse the faulty module with small performance penalty.

Type 2: Fault in one of the electrodes adjacent to the corner electrode

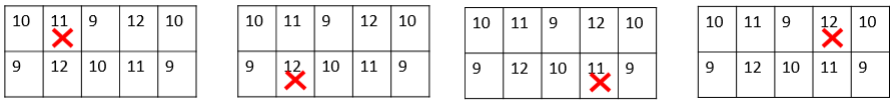

Figure 3.9: Faults in electrodes adjacent to the corner electrode

For example, electrode 11 next to the corner cell 10 is considered to be faulty. In such a case, the droplet can be routed as shown in Fig 3.10. Owing to the symmetry in the module, the path taken by the droplet will be the same for all Type 2 faults shown in Fig 3.10.

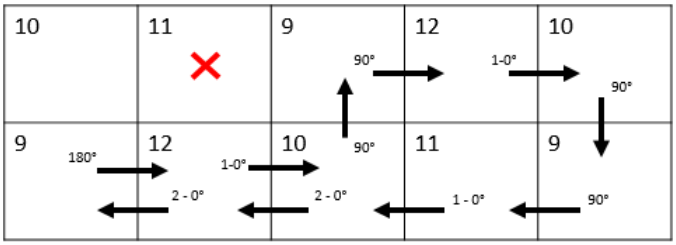

Figure 3.10: Mixing route for *Type 2* fault

Each rotation can be decomposed into single steps as labelled in Fig 3.10. The routing path is made up of 3 steps in 1- 0°, 2 steps in 2 - 0°, 4 steps in 90° and 1 step in 180°. The

mixing percentage completed in 1 rotation can be calculated as follows.

$$\begin{aligned}\text{Percentage completed in 1 rotation} &= 1\times\ (\text{-}0.52) + 4 \times (0.1) + 3 \times (0.29) + 2 \times (0.89)\\ &= -0.52 + 0.4 + 0.87 + 1.78\\ &= 2.53\%\text{mixing per rotation}\end{aligned}$$

The total rotations needed to complete the operation is equal to 100/2.53 i.e., 39.53 rotations. The number of steps in each rotation is 10, so the time taken for each rotation is 0.10 seconds. Therefore, the total mixing time is 3.953 seconds. Some operations can be scheduled on this reconfigured module rather than completely discarding it. The mixing time of a 2×3 module is 6.1 seconds. Thus by reconfiguring the module as shown in Fig 3.10 we get a mixing time of 3.95 seconds which is 36% better than the mixing time of a 2×3 module.

Type 3: Fault in electrodes in the middle column

Figure 3.11: Faults in electrodes in the middle column

In this case, let us assume that the electrode labelled 9 in the middle column is faulty. In the presence of such a fault the droplet can be routed as shown in Fig 3.11. For the fault on the middle electrode 10 as shown in Fig 3.12, the mixing time can be derived similarly.

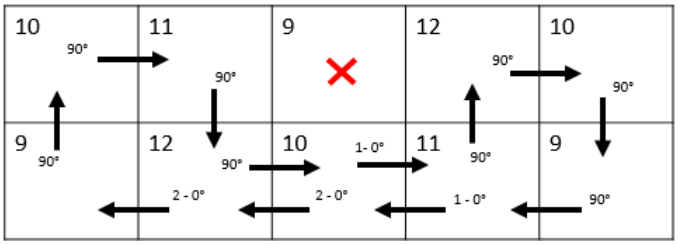

Figure 3.12: Mixing route for *Type 3* fault

Each rotation can be decomposed into single steps as labelled in Fig 3.12. The routing path is made up of 2 steps in 1 - 0°, 2 steps in 2 - 0° and 8 steps in 90°. The mixing percentage completed in 1 rotation can be calculated as follows.

$$\begin{aligned}\text{Percentage completed in 1 rotation} &= 8\times\ (0.1) + 2\ \times\ (0.29) + 2\ \times\ (0.89) \\ &= 0.4 + 1.45 + 1.74 \\ &= 3.16\%\ \text{mixing per rotation}\end{aligned}$$

The total number of rotations needed to complete the operation is 100/3.16 i.e., 31.64. The number of steps in each rotation is 12, so the time taken for each rotation is 0.12 seconds. Therefore, the total mixing time is equal to 3.797 seconds. Thus, by reconfiguring the module rather than completely discarding it, some operations can be scheduled on this reconfigured module with reasonable completion time.

3.2.2 Multiple Electrode Faults

Multiple electrode faults can result either from multiple single electrode failures as discussed above or from failures in the scan and hold flip-flops in the scan chains used to control the electrodes. Let us assume failures are in the hold part of the scan and hold flip-flop, and this will cause failures in all electrodes connected to this particular scan and hold flip-flop.

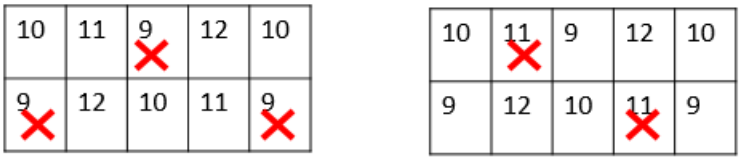

Figure 3.13: Multiple electrode failure

Failures in the hold flip-flop will cause all electrodes connected to that particular flip-flop to fail. This can result in two types of failures as shown in Fig 3.13. Reconfiguration for these type of failures is discussed below.

Type 4: Failure on 3 electrodes (e.g. Electrodes labelled 9 or 10)

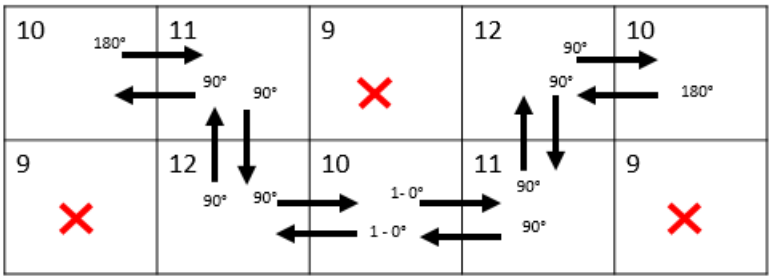

Figure 3.14: Mixing route for *Type 4* fault

Fig 3.14 demonstrates a path the droplet can take in the presence of Type 4 multiple electrode faults. Owing to the symmetry, the path taken by the droplet will be similar in case of failures on electrodes labelled 10. The routing path in Fig 3.14 is made up of 2 steps in 1 - 0°, 8 steps in 90° and 2 steps in 180°. The mixing percentage completed in 1 rotation can be calculated as follows.

$$\begin{aligned}\text{Percentage completed in 1 rotation} &= 2\times\ 0.29 + 8\ \times\ 0.1 + 2\ \times\ \text{(-0.52)} \\ &= 0.58 + 0.8 - 1.04 \\ &= 0.34\%\ \text{mixing per rotation}\end{aligned}$$

The percentage of mixing completed in 1 rotation is 0.34% and the total number of rotations required for operation completion is 100/0.34 i.e. 294.117 rotations. Since the number of steps in each rotation is 12, the time taken for each rotation is 0.12 seconds. Therefore the total mixing time is 35.29 seconds.

The mixing time is prohibitively large when compared with the mixing time of a 2×5 module (2.202 seconds). Scheduling an operation on this reconfigured module will block the execution of children nodes for a long time. In such a case it makes more sense to discard a module with multiple electrode failures like in Fig 3.14.

Type 5: Failure on 2 electrodes (e.g. Electrodes 11 or 12)

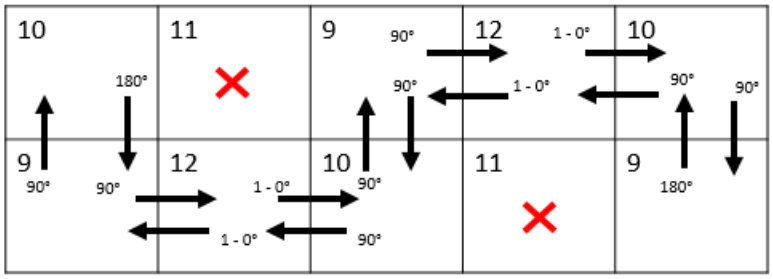

Figure 3.15: Mixing route for *Type 5* fault

Fig 3.15 demonstrates a path the droplet can take in the presence of *Type 5* fault. Owing to the symmetry, the path taken will be similar to the case of failures on electrodes labelled 12. Each rotation can be decomposed into single steps as labelled in Fig 3.15. The routing path in Fig 3.15 is made up of 4 steps in 1 - 0°, 8 steps in 90° and 2 steps in 180° .The mixing percentage completed in 1 rotation can be calculated as follows.

$$\begin{aligned}\text{Percentage completed in 1 rotation} &= 4\times\ 0.29 + 8 \times 0.1 + 2 \times \text{(-0.52)} \\ &= 1.16 + 0.8 - 1.04 \\ &= 0.94\% \text{ mixing per rotation}\end{aligned}$$

The percentage of mixing completed in 1 rotation is 0.94%, and the number of rotations for operation completion equal to 100/0.94, i.e., 106.38. Since the number of steps in each rotation is 14, the time taken for each rotation is 0.14 seconds. Therefore the total mixing time is 14.89 seconds. Similar to the Type 4 failures, the mixing time in this case is also too large. Therefore a module with this type of failures should be discarded.

3.3 Fault Tolerant synthesis for GPFP architecture

For application/assay specific DMFBs, electrode actuations are determined during the design process and are stored in a controller connected to the array. In the case of general purpose DMFBs, the controller is programmed on the fly according to the assay that will be executed on it. The process of generating electrode actuation patterns for an assay based on

parameters like DMFB array used, expected completion time etc. is called *synthesis*. The target assay to be executed is modelled as a directed acyclic graph (DAG), with its nodes denoting the operations to be performed in the assay and the edges signifying dependencies between them. The DMFB array is modelled as an array of electrodes with various input output ports and peripherals like detector, heater etc. In the case of the GPFP architecture, the modules and pin-constraints are modelled according to the architecture specified in Fig 2.4.

Once the array is modelled, we select modules from the library which we intend to use on this array. This step is called *allocation*. In the case of the GPFP architecture, since all modules are configured as 2×5 mixers, we skip this step. We then move on to *scheduling*, which determines the start and stop time of each operation in the DAG, considering dependencies between operations and resource constraints. After *scheduling*, the next step is *binding* which binds each operation scheduled in a particular time step to its corresponding module. The next step is *placement* in which we place these modules in each time step on the DMFB array such that there is no overlap. After *placement*, we need to route droplets between different modules based on operation requirements, this phase is called *routing*. In the GPFP architecture, the modules are pre-placed and therefore the binding and placement steps are combined into one single binding step.

In [8] the author proposed that any scheduling algorithm which has scheduling results comparable to the best available algorithms or which has small execution time is a good fit for the GPFP synthesis. To evaluate the performance of the GPFP architecture, they use the well-known *List Scheduling* algorithm. List scheduling is a greedy scheduling algorithm, and the psuedo code for the algorithm is explained in Algorithm 1.

A straightforward way to add fault tolerance to the existing scheduling algorithm is to update the list of modules, *Lmodules*, such that we can simulate a faulty module and a reconfigured module. We assume an offline fault detection approach. Prior to executing any assay, we run the testing process to determine location of faults if there is any. Based

```
Data: assay specified as a DAG, Architecture, faults
Result: Scheduled assay
Read in list of nodes to execute and store them in list Loperations;
Read in the list of modules on the array and store them in list Lmodules;
Set current time step to zero;
Create a list Lready to hold nodes that are ready to execute;
if faults specified then
    Update the faulty module in list Lmodules;
end
while all operations in Loperations are NOT complete do
    for each node in Lnodes do
        if node status is not COMPLETE/EXECUTING & parent nodes are
        COMPLETE then
            Add node to Lready.;
        end
    end
    for each node in Lready do
        if modules availabe in Lmodules then
            Set this module as NOT available;
            Set current node as executing;
            Set node start time as current time;
            Set node complete time as time taken by this module to complete
            operation;
            Set module free time as time taken for node to complete;
        end
    end
    Increment current time step. ;
    for each module in Lmodules do
        if module is NOT available AND module free time equal to current time step
        then
            Mark node executing on this module as COMPLETE;
            Set module as AVAILABLE;
            Remove node from list Lready;
        end
    end
end
```

Algorithm 1: List Scheduling algorithm for Fault tolerant GPFP architecture

on this information, we reconfigure the faulty module accordingly. For a good 2×5 module in the GPFP architecture, the completion time for mixing is 2.2 seconds. To simulate an array with a single faulty module we discard that module by reducing the number of available modules in *Lmodules*. To simulate a reconfigured module, we add a new module to *Lmodules* depending on the fault type and mixing time summarized in Table 3.3. The mixing time required depends on the fault targeted during the experiment. The target fault is passed to the modified scheduling algorithm. Based on this value, an appropriate reconfigured module is selected and added to *Lmodules*.

Table 3.3: Mixing time of reconfigured modules based on fault type

FAULT TYPE	**MIXING TIME(in seconds)**
Type 1	2.9
Type 2	3.953
Type 3	3.797

We now explain the modified list scheduling shown in Algorithm 1 with an example. Fig 3.16 shows the DMFB architecture used on the right and assay that is executed on the left. The DMFB in Fig 3.16 has four dispense ports which hold the samples/reagents used in the assay, one output port and two 2×5 general purpose modules. We read in the DMFB input file and store the information of non-reconfigurable modules, i.e., dispense ports, output ports and reconfigurable modules like 2×5 modules in this case. Assuming no faults, we add both two 2×5 modules to a list *Lmodules*. In the case of a faulty module, we assume that the faulty module is discarded and remove one module from *Lmodules*. To simulate a reconfigured module, we add a new module to the list *Lmodules* according to the type of the fault (single fault assumption). We read in the assay description file and store all the nodes as a directed acyclic graph (DAG), and then add the nodes to list called *Lnodes*. We set the current time step as zero and begin the execution of Algorithm 1.

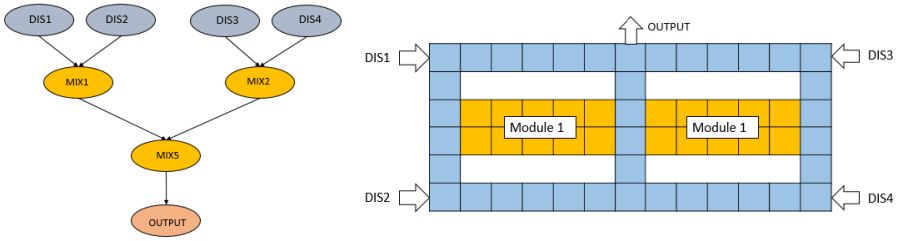

Figure 3.16: Example assay & DMFB on which it is executed

While all nodes in the assay are not marked as complete, we check the list *Lnodes* for nodes that are ready to be executed. At time zero, since dispense ports have no parent operations, we set all dispense operations (DIS1, DIS2, DIS3, DIS4) as ready to be executed and add them to *Lready*. After adding all nodes that are currently ready to execute to the *Lready* list, we go to the next step. In this step we iterate over *Lready* and execute nodes according to the availability of all resources. If the required resource is available, we mark the current node as executing and update its start and end time. After iterating over the *Lready* list, we increment the timestep and check for operations that are complete. Since dispensing operation is assumed to take a single time step, we mark dispensing operation as complete.

In the next iteration of the while loop, MIX1 and MIX2 are added to the ready list and executed depending on the availability of modules. We continue this until all nodes in the assay are marked as complete.

3.4 Results

The DMFB Static Synthesis Simulator (DSSS), is an open source simulation platform developed by Grissom et al. [36]. The platform is modular and flexible and it is easy to develop new architectures and algorithms and compare them to existing algorithms without too much effort. We use a modified version of the DSS presented in [8]. The DSS simulator is modified to support the GPFP architecture. To add fault tolerant capabilities to the

GPFP architecture, we modified the scheduling and the binding algorithm as discussed in the previous section.

We compare the assay completion time on the GPFP architecture with no faults, with one faulty module (which covers all types of faults discussed earlier) and reconfigured faulty module conditions. We use the benchmarks used in [8] and [36] to evaluate the fault tolerance algorithm. We simulate the polymerase chain reaction (PCR) assay which is used to amplify or quantify DNA or RNA. We simulate a convergent PCR assay. In-vitro assays are used to measure glucose and other metabolites in blood, plasma, urine and saliva. This is important in clinical diagnosis as variation in these metabolites is often a symptom of various disorders.

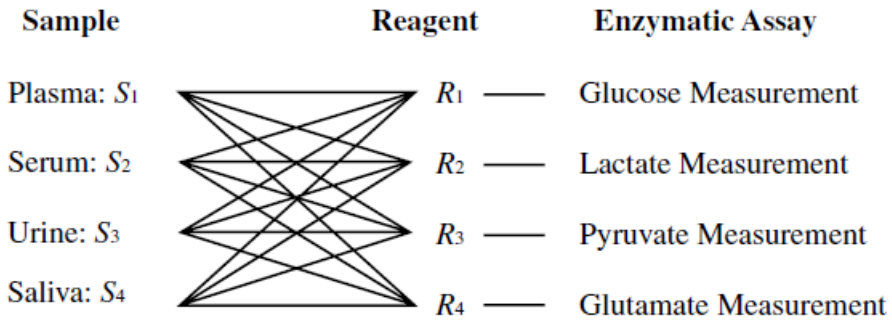

Figure 3.17: Schematic of in-vitro diagnostic assay

We implement various combinations of in-vitro diagnostics (Fig 3.17), measuring multiple metabolites thereby increasing the number of operations performed on the array. We also implement protein split assays (Fig 3.18) which are divergent assays. The numerical value in the name of the protein split assay specifies the number of split levels in the assay as shown in Table 3.4. As shown in Fig 3.18, as the number of split levels increases, the number of operations and thereby the complexity of the assay increases.

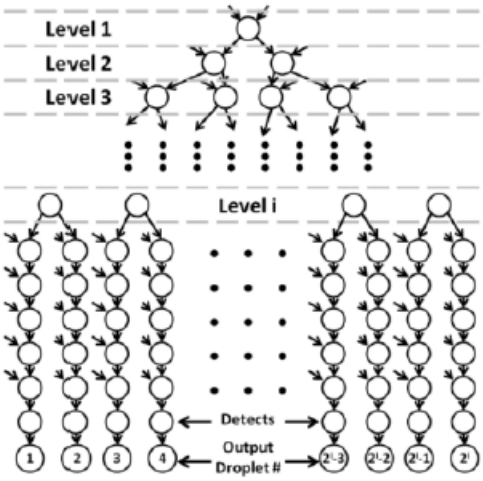

Figure 3.18: Schematic of protein Split assay

We assume that the droplet actuation frequency is 100Hz, i.e., the time step is 0.01 seconds. Similar to the experiment in [8], we do not consider the routing time required for each assay in these results due to its small value. These simulations were run on a 2nd generation Intel i5 processor with 12 GB of RAM running 64-bit Windows 7. The simulation results are summarized in Table 3.4.

Table 3.4: Mixing time of reconfigured modules based on fault type

Assay	Dimensions	Modules	No Fault	With Fault	Type -1	Type - 2	Type -3
B1_PCR	13×9	4	6609	8811	7307	8360	8204
B1_PCR	13×12	6	6609	6609	-	-	-
B2_InVitro_Ex1_2s_2r	13× 9	4	4409	8811	5805	7911	7599
B2_InVitro_Ex1_2s_2r	13×12	6	4409	4409	-	-	-
B2_InVitro_Ex2_2s_3r	13× 9	4	8813	8813	-	-	-
B2_InVitro_Ex2_2s_3r	13×12	6	4413	8811	5809	7915	7603
B2_InVitro_Ex3_3s_3r	13× 9	4	13215	13217	13215	13215	13215
B2_InVitro_Ex3_3s_3r	13×12	6	8813	8813	-	-	-
B2_InVitro_Ex4_3s_4r	13× 9	4	13219	17619	16707	17619	17619
B2_InVitro_Ex4_3s_4r	13×12	6	8813	13217	11605	14066	13598
B2_InVitro_Ex5_4s_4r	13× 9	4	17619	26427	22505	22023	22023
B2_InVitro_Ex5_4s_4r	13×12	6	13217	17619	13217	14066	13598
B4_ProteinSplit_1_Eq	13× 9	4	15417	15417	-	-	-
B4_ProteinSplit_1_Eq	13×12	6	15417	15417	-	-	-
B4_ProteinSplit_2_Eq	13× 12	6	17621	30831	21111	24625	24001
B4_ProteinSplit_2_Eq	13×15	8	17621	17621	-	-	-
B4_ProteinSplit_3_Eq	13× 12	6	33035	46245	33035	33035	33035
B4_ProteinSplit_3_Eq	13×15	8	19827	33033	23317	26831	26207
B4_ProteinSplit_4_Eq	13×18	10	48449	61659	48449	48449	48449
B4_ProteinSplit_4_Eq	13×21	12	35241	35243	35241	35241	35241

Let us analyze the result in the 1st row, the PCR assay executing on the 13×9 array, which has 4 modules as shown in the Fig 3.19. The PCR assay shown in Fig 3.20 has 4 operations that should be executed and completed in parallel to enable the children operations. Operations MIX 3, 6, 9 and 12 should be executed in parallel to reduce the assay completion time. In the case of a fault in the 13×9 array, the number of available modules is 3, therefore only 3 of those parallel mixing operations can be scheduled. This in turn increases the assay

completion time to 8811 milliseconds from 6609 milliseconds as shown in Table 3.4.

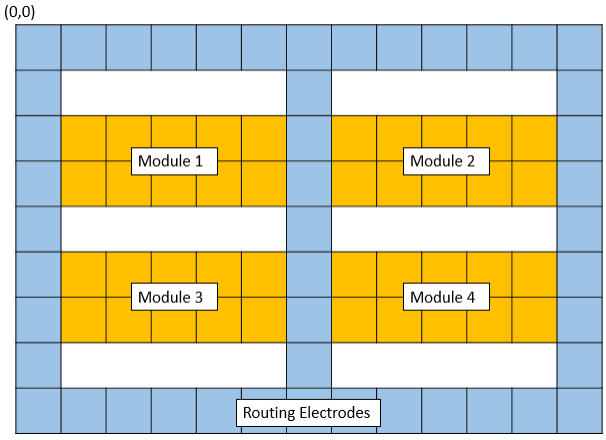

Figure 3.19: Modules on a 13×9 GPFP array

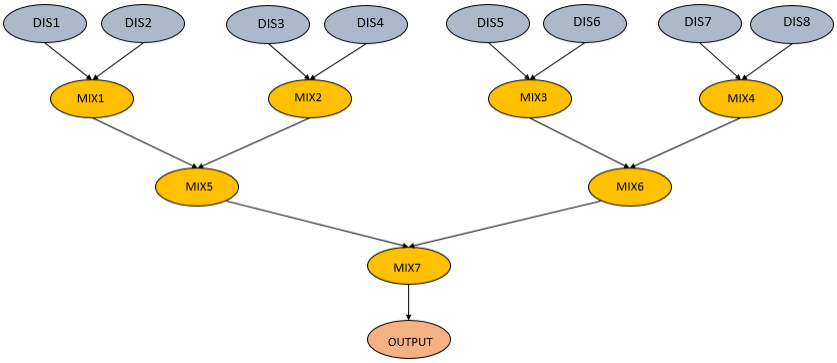

Figure 3.20: PCR Assay

By reconfiguring the faulty module, we have a slower 4th module, which can help the parallel mixing operation to finish at a slower rate. We simulate all single electrode faults namely Type 1, 2 and 3. We observe that the effect of a faulty module is dependent on the number of parallel operations in the assay, the number of modules available on the array, and the type of fault. In the case where the number of available modules is greater than the number of parallel operations, for e.g., row 2 in Table 3.4, the presence of a faulty module has no impact on the assay completion time. We observe an average improvement of 31.62% in completion time of the assay in the case of Type 1 fault, 17.18% in the case of Type 2 fault and 19.67% in the case of Type 3 fault.

By reconfiguring modules with routing-based operation execution we are able to improve the assay completion time for arrays with faulty modules. In the next chapter we will apply the routing-based operation execution results to develop a different synthesis algorithm.

Chapter 4

Routing Based Synthesis

In chapter 3, we characterized the percentage of mixing completed when the droplet moves in a particular direction. In this chapter, we apply this routing-based methodology to develop a different type of synthesis mechanism. The previous work on synthesis assumes the presence of virtual modules to complete reconfigurable operations like mixing, splitting etc. Since these operations can be performed by routing the droplets on any random sequence of electrodes, we do not need a virtual bounding box. In this chapter, we first discuss the limitations of this module-based synthesis approach and then propose a routing-based synthesis approach to overcome these limitations.

4.1 Motivation for routing-based synthesis

Let us consider the synthesis of the example assay shown in Fig 4.1 on a 7×7 array. The example assay has seven input/dispense operations (D1 - D7), six mixing operations (M1 - M6) and one output operation (O). Let us assume that we have a 2×4 mixer with mixing time of 2.9 seconds and a 2×5 mixer with mixing time of 2.2 seconds.

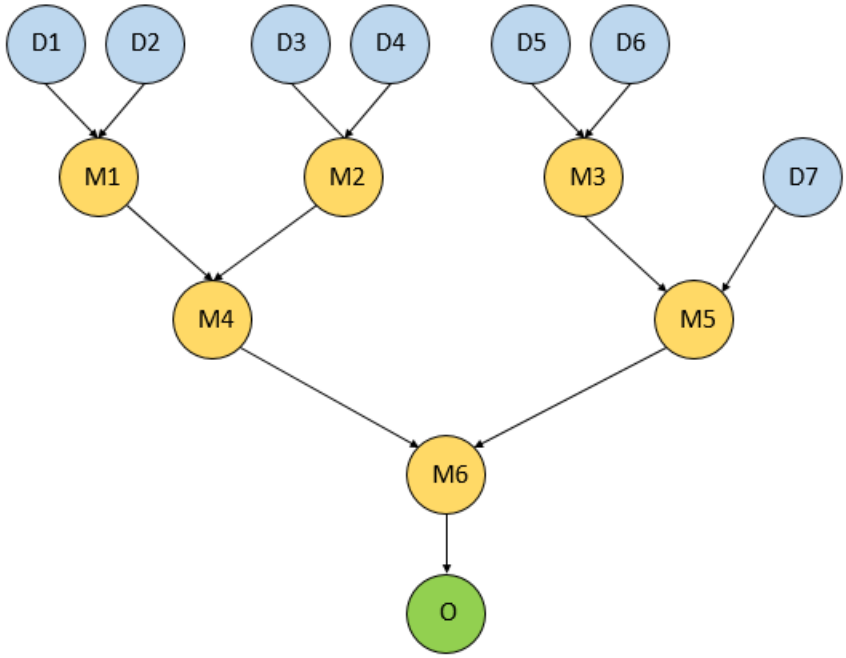

Figure 4.1: Example assay

For simplicity, we assume that dispense operations D1 to D4 are complete and their children nodes are ready to execute. The droplets are dispensed and are ready on the chip for merging. We ignore the overhead of routing the droplets, since the time taken to route droplets is too small compared to the mixing time. Since each merging operation is essentially routing two droplets to a common location, merging time can be ignored as well. The optimal solution for this assay is shown in Fig 4.2. The placement for this schedule is as shown Fig 4.3.

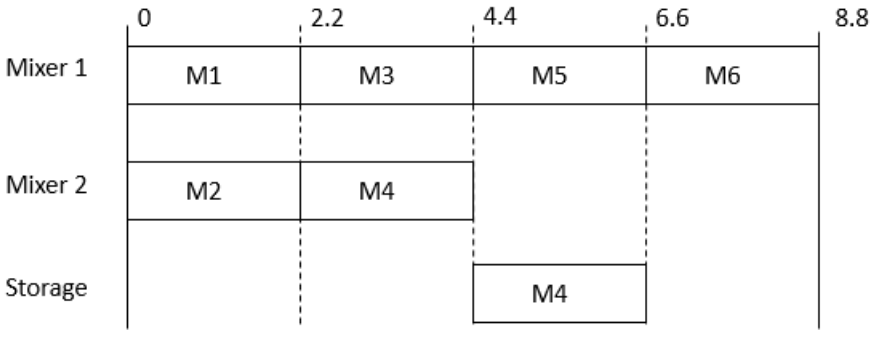

Figure 4.2: Schedule for the example assay

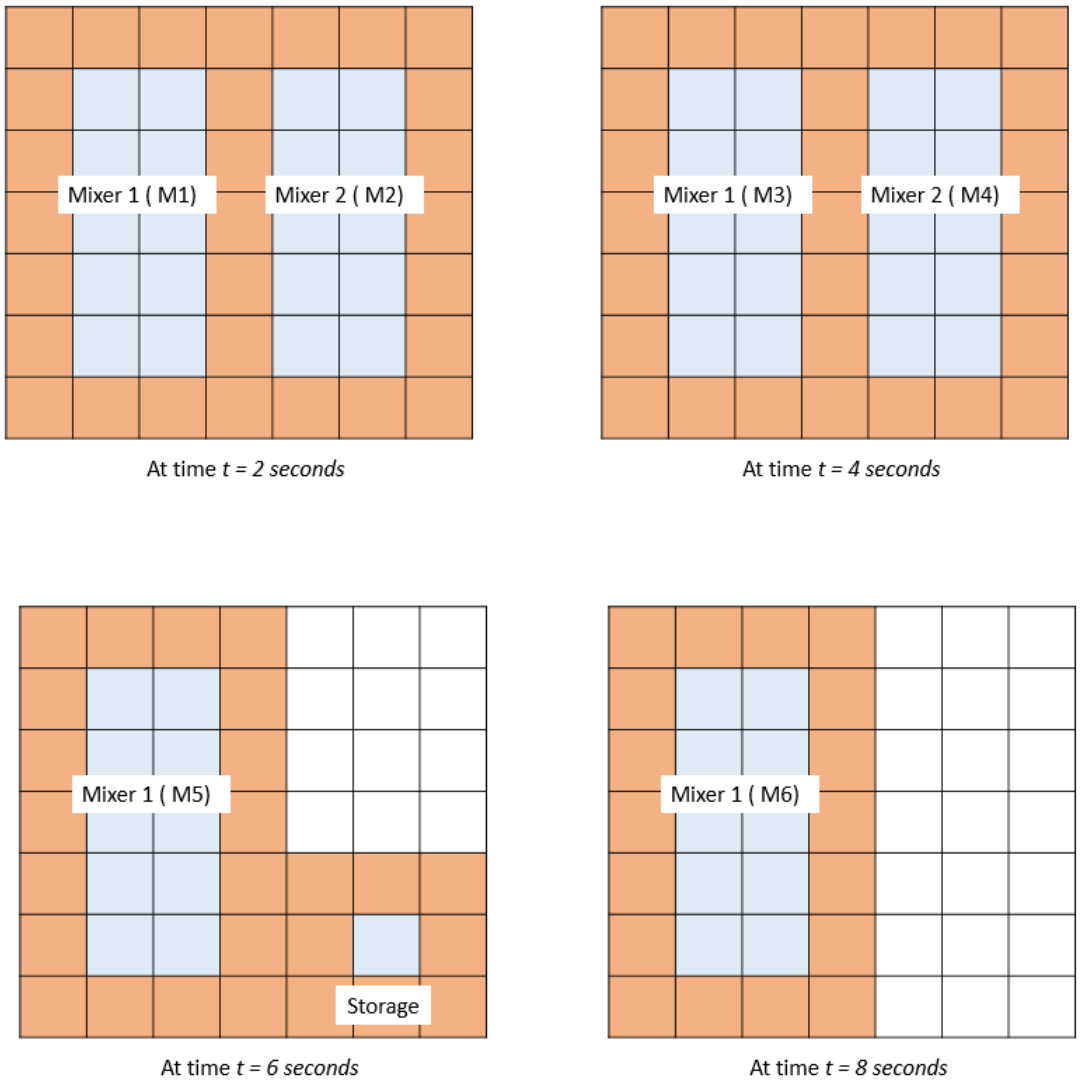

Figure 4.3: Placement for example assay

The module-based synthesis approach considers all electrodes (colored blue in Fig 4.3) belonging to a module as occupied during operation execution. In addition to that, to maintain fluidic constraints, i.e., to prevent accidental merging or mixing of droplets, a segregation layer is maintained around the modules (electrodes colored in red in Fig 4.3). Some module placement algorithms allow segregation layers to overlap. For the 7×7 array, only two 2×5 modules can be placed at a time, and therefore we can only have two operations execute at a time. In our example assay, we have three operations that can be executed in parallel. Using smaller modules (For example, 2×2 having mixing time of 9.1 seconds)

will increase the assay completion time drastically. Therefore, we have to schedule mixing operation M3 in the next time step. Let us consider the 2×5 modules in Fig 4.3. This module is made up of 10 electrodes that perform mixing and 18 electrodes that are a part of the segregation layer, i.e., 28 electrodes in all are considered to be occupied when an operation is being executed on this module. Since a droplet occupies only one electrode at any given time, the remaining electrodes can be used by other droplets to complete their operations. This allows multiple operations to be scheduled at a time, thereby increasing the parallelism in operation execution by better utilization of electrodes on the array.

The schedule for the example assay in Fig 4.1 using a routing-based synthesis approach is shown in Fig 4.4. We assume that all dispensing operations are complete and the droplets are on the array ready for the next operation. The time taken to route droplets around is negligible compared to the time taken for a mixing operation to complete therefore, we ignore the routing time. Mixing using the routing based-technique takes an average of 2 seconds to complete depending on the size of array and the number of droplets on the chip. We therefore assume 2 seconds as the upper bound for all mixing operations in the schedule shown in Fig 4.4.

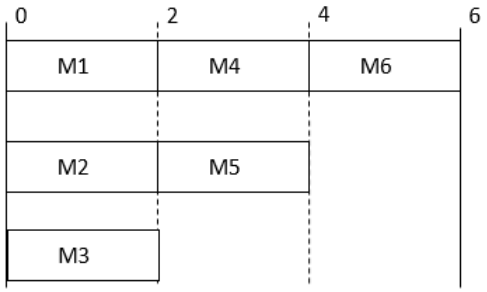

Figure 4.4: Example assay schedule using RBS

Operations whose parents nodes are completed are ready to execute. Once dispensing operations D1 to D6 are complete, mixing operations M1, M2 and M3 are ready to execute. The dispensed droplets are merged to form mix droplets M1, M2 and M3 and the mixing operations begin. In each step the droplets are routed based on the mixing values mentioned

in Table 3.2 such that the greatest mixing percentage can be obtained. The movement of the droplets at intermediate time steps during the execution of example assay is shown in Fig 4.5. The routing-based synthesis approach offers much better parallelism compared to the module-based synthesis approach as shown in Fig 4.2 and Fig 4.4.

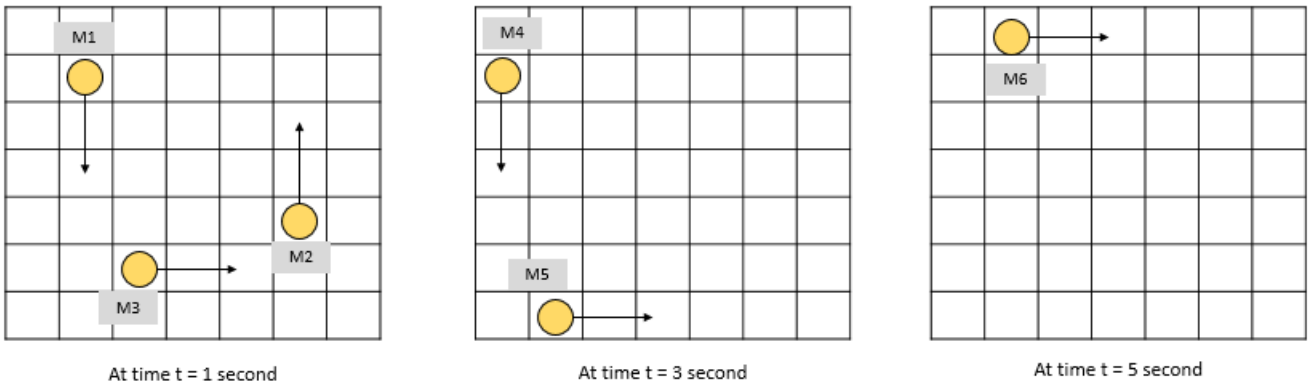

Figure 4.5: Movement of droplets during execution

The impact of electrode failures is significant on module-based synthesis. During reconfiguration the modules have to be placed such that they do not overlap the faulty electrode. As a result, at times we may have to use smaller modules which impacts the assay completion time. As shown in Fig 4.6, on the left we have to use a smaller 2×2 mixer in presence of a faulty electrode. In the case of routing-based approach, the droplets can be routed around the faulty electrode. Thus, the routing based-synthesis approach has better fault tolerance.

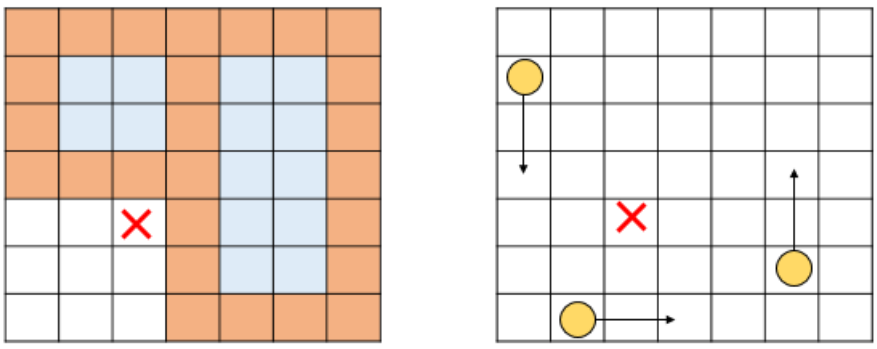

Figure 4.6: Effect of faults on module based and routing based approach

The module-based synthesis method is made up of steps like allocation, binding, schedul-

ing, placement and routing. The computational complexity of each of those steps is high. To obtain an optimal solution, we frequently use heuristics like Tabu search etc. to iterate over the search domain. This further increases the complexity of the module-based synthesis approach. Routing-based synthesis transforms the synthesis problem into a routing-problem which is a simple deterministic selection problem.

4.2 Routing based synthesis algorithm

The disadvantages of module-based synthesis approach based on virtual modules is discussed in the previous section. In this section we propose a routing-based synthesis algorithm in which operations are executed by routing the droplets on a random sequence of electrodes on the array. In this work, we propose a greedy algorithm which executes operations by routing the droplet along the best possible path at every time step. The major difference between this algorithm and the routing-based algorithm presented in [37] is that this algorithm uses a deterministic approach in selecting the next step for the droplet and it has a better merging algorithm. The following pseudo-code gives a high level overview of the algorithm.

Data: assay specified as a DAG, Architecture;
Result: Scheduled assay
Read in list of nodes to execute and store them in list *Lnodes*;
Set current time step to zero;
Create a list *LcompletedNodes* to hold nodes that are complete;
while *Length of Lnodes is NOT EQUAL to Length of LcompletedNodes* **do**
    Call addReadyNodes();
    Call executeReadyNodes();
    Call bookkeeping();
end

Algorithm 2: Routing based sythesis psuedo code

The inputs to the routing-based synthesis (RBS) algorithm are: (a) the array on which an assay is to be scheduled and (b) the assay in form of nodes and dependencies between them represented by the edges. The architecture input file is read in and the details of the architecture such as its dimensions, location of the input output ports and peripherals such

as detectors, heaters etc. are stored. A directed acyclic graph (DAG) is constructed based on the assay input file. Each node of the DAG represents operations to be executed and the edges define the dependencies between nodes. A list of nodes *Lnodes* is constructed after reading the DAG. This list contains all nodes/operations to be executed for this assay. We construct another list of nodes *LcompletedNodes*, which is empty initially. As nodes are completed, they are added to this list. While all the nodes are not executed i.e., the length of *Lnodes* is not equal to the length of *LcompletedNodes*, we execute the following steps.

STEP 1: addReadyNodes

This step uses the well-known List Scheduling method to schedule operations for execution. We scan through the list *Lnodes*, for each node that is not complete/executing/dispense, we check the status of its parent nodes. If the parent node(s) is (are):

- Complete, the node is ready for execution and we add it to the *LreadyNodes* list.
- If one of the parent node is complete, and the other parent is dispense, we add the dispense parent to the *LreadyNodes* list.
- If both parent nodes are dispense, add parent dispense operations to the *LreadyNodes* list.
- Otherwise, continue to the next node.

The function addReadyNodes returns after iterating through all nodes in the list. *LreadyNodes* is updated with all nodes that are ready to be executed at that particular time step.

STEP 2: executeReadyNodes

In this step, we iterate over the list *LreadyNodes* created in the previous step. This step executes all operations that are ready to be executed in the current time step. Each node is handled based on the type of operation it belongs to. If the node status is READY, its start time is updated as the current time step and its status is set as EXECUTING. If the

node has its status as EXECUTING, it is currently being executed, we continue executing it at each time step until the operation is complete. The different operations than can be performed in this step are as follows,

1. *Dispense Droplet* : In this operation, a new droplet is dispensed from one of the input reservoirs and is added to the DMFB array. We check if the new droplet can be dispensed from the port. The presence of another droplet around the port violating fluidic constraints may block the dispensing operation. This is shown in the Fig 4.7.

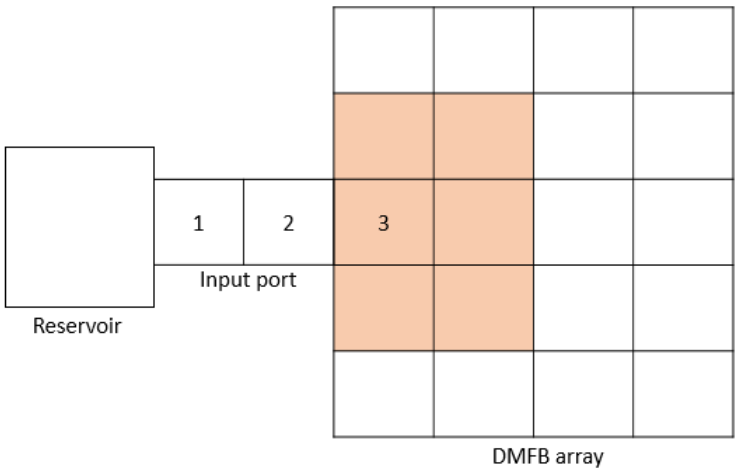

Figure 4.7: Dispense operation

If there is another droplet in the shaded region, the dispensed droplet may accidentally merge with it. To avoid this we dispense a new droplet only when there is no other droplet in the shaded region. According to the module library mentioned in [37], dispensing operation takes 2 seconds, but considering Fig 4.7, dispensing operation is completed by moving the droplet over 3 electrodes. The droplet can be dispensed from the reservoir by activating electrodes 1, 2 and 3 in that order. Since the time taken for a droplet to move from one electrode to the other is 0.01 seconds, the total time for dispensing operation to complete will be 0.03 seconds. In this work we assume that the dispense droplet is ready at electrode 2 in the previous time step and the time taken to dispense a droplet is 0.01 seconds. Once a new droplet is available at electrode 3,

we mark the executing node as COMPLETE.

2. *Mix Droplet*: A mixing operation is denoted as shown in Fig 4.8. Let us assume that droplet D1 is created for node DISPENSE_1 and droplet D2 is created for node DISPENSE_2. For mixing operation MIX_3 to begin, droplets D1 and D2 must merge to create a new droplet, droplet D3. Thus, the mixing operation is broken down into 2 parts, first merging the droplets and then mixing them.

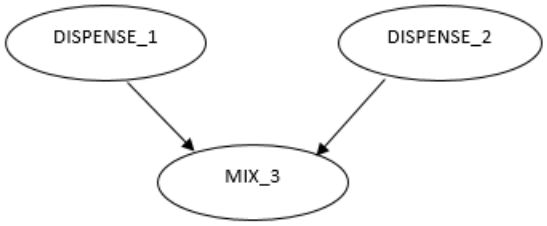

Figure 4.8: Mixing operation

When the operation being executed is mixing, we check if there exists a droplet associated with the mixing node (for example droplet D3 associated with node MIX_3 in Fig 4.8) on the array, if not the parent droplets have not merged yet to form the mixing droplet. In such a case the parent droplets must be merged. In [37], the authors use Manhattan distance between the two droplets to determine the best possible direction the droplet must move in. Out of the five possible directions in which any droplet can move at any instant, the author generates a list of the best three directions the droplet can move and randomly selects one of those. This randomized approach may or may not select the best possible move at that instant.

We use a deterministic approach to determine the direction in which both droplets should move in order to merge. We apply the well known Lee's maze routing algorithm to determine the best possible path for the droplets. This approach considers the presence of all other droplets on the array which may block paths suggested by calculating the Manhattan distance between both droplets S and T in Fig 4.9.

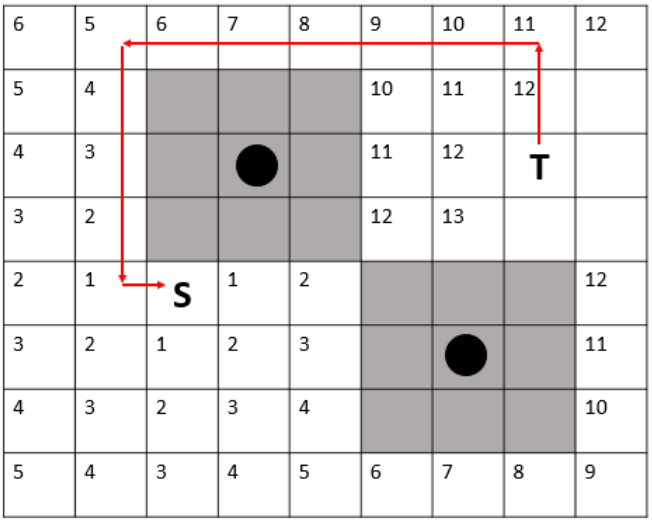

Figure 4.9: Droplet merging using Lee's Algorithm

Let us merge the droplets at locations denoted by S and T in Fig 4.9. If we consider moving the droplets such that the Manhattan distance between them is reduced, we may be blocked by other droplets. The shaded region around the droplet denotes the segregation layer around them to maintain fluidic constraints. In such a case, we may need to move initially in directions that may increase the Manhattan distance between these two droplets. The Lee's algorithm helps us find the route even in the presence of other droplets. We begin by marking the droplets as Source (S) and Target (T). We then propagate a wave of numbers, like in Fig 4.9 and each number denotes the number of steps droplet at S needs to take to reach that electrode. We stop once we reach the target T. We then backtrack from T, tracing the numbers in decreasing order until we reach S. We use this backtracked path to determine the direction in which both droplets S and T should move for merging. We continue this operation at every time step until both droplets merge. Once the droplets are merged we begin with mixing.

The mixing operation can start once the parent droplets merge to form a mix droplet. We use a greedy approach to determine the best direction in which the droplet should move in order to complete mixing as soon as possible. For a newly merged droplet, based on its current location, we move it such that it can continue its linear movement

as long as possible. We assume that this step completes 0.29% of the mixing. We record the entire route taken by the droplet and use it to calculate the new mixing percentage in the current time step. If the droplet continues moving in the current direction for more than one electrode (2-0°), we add 0.89% to the mixing percentage; if it is just one electrode in the current direction (1-0°), we add 0.29% to the mixing percentage. If the droplet cannot continue its movement in the current direction we move the droplet 90° of the current direction such that the droplet can continue its straight line motion for the maximum number of electrodes. After moving the droplet 90° of the current direction we add 0.1% to the mixing percentage. If the droplet cannot be moved 90° as well, we keep the droplet at its current location. This does not change the mixing percentage. We continue moving the droplet at each time step until the mixing percentage is 100%.

3. *Split droplet*: The next operation we consider is to splitting a droplet. To split a droplet, we have to simultaneously activate the electrode on either side of the droplet, either in X direction or in Y direction. We first check the location of the droplet. If it is located along the boundary of the array, we can only split it in either in X direction or Y direction; otherwise we try splitting in X direction first, if not possible we try Y direction. The droplet can be split, only if both new droplets can be placed on the array without violating any fluidic constraints. If this is not possible, we move the droplet in a randomly selected direction so that we can split the droplet in the next time step. Since splitting operation involves splitting a droplet and moving one electrode, we assume that the time taken for splitting operation is 0.01 seconds.

4. *Detect operation*: For detect operation we move the droplet to the electrode marked as detector. We use the Lee's algorithm to find the best path for the droplet to move to the detector. The droplet moves one electrode at a time. We continue this until the droplet reaches the detector electrode. Once the droplet reaches the detector electrode

we mark the detect operation as COMPLETE. The time taken for detect operation to complete is equal to the time taken for the droplet to move the detect electrode.

5. *Output operation*: For output operation to complete we move the droplet to the output port. Like in detect operation, we use the Lee's algorithm to find the best path to move the droplet to the output port. The droplet moves one electrode at a time. Once the droplet reaches the output port we mark the operation as COMPLETE. Similar to detect, the time taken for output operation equal to the time taken by the droplet to reach the output port.

STEP 3: bookkeeping

In this step, we go through the list of ready nodes *LreadyNodes* and check the status of each node. We continue to the next node if the status is READY or EXECUTING. If the status of the node is COMPLETE, we remove it from *LreadyNodes* and add it to the list of completed nodes *LcompletedNodes*. The MIX operation droplets are handled differently.

For a MIX operation, first we check if the mixing percentage is 100%, if not the operation is still EXECUTING. If the mixing percentage is 100%, we check the location of the droplet. Once we set the mix operation as complete the droplet remains at the current electrode until its child operation is ready to execute. In most cases, the mix droplet combines with a new droplet from the dispense port for next operation. If the MIX droplet is around the periphery of the array, the dispense operation is blocked thereby causing a deadlock situation. To avoid this, once the mixing percentage reaches 100%, we check the location of the droplet. If it is located close to the periphery it is moved closer to the center of the array in the next time step. Otherwise, the operation is marked as COMPLETE and removed from the list of ready nodes. We then add this operation to the list of completed nodes *LcompletedNodes*. After iterating over all nodes in the *LreadyNodes* list we increment the current time step by one.

DEADLOCK AVOIDANCE: We keep track of each droplet at every time step. If the droplet is at its current position for more than 3 time steps, we assume that the droplet is in

a deadlock situation. We then move the droplet randomly in any possible direction to break the deadlock. If the droplet is a mix droplet, we update the mix percentage appropriately.

4.3 Assumptions

Architectural assumption: We assume that the RBS (routing based synthesis) algorithm is executed to schedule an assay on a general purpose DMFB, i.e., an array of electrodes with no architectural modifications like the FPPC/GPFP architecture and no pin-constrained designs. Pin-constrained architectures have been proposed as a solution to the growing complexity of direct addressing (DA) arrays as the number of electrodes on the array increased. In [25], the authors propose an active matrix addressing (AMA) approach, in which m + n pins can be used to control an m $\times$ n electrode array. We assume that the electrodes are individually addressable using either DA or AMA techniques.

Dispensing time: According to [37], the time taken for dispensing operation to complete is 2 seconds. According to Fig 4.7, dispensing operation needs a droplet split and move operation and the time taken for this to complete is 0.03 seconds. We assume that dispense droplet is ready on electrode 2, hence the operation is instantaneous and needs one time step for completion. Once the assay is scheduled an electrode activation sequence file is generated, it can be modified such that Electrodes 1 and 2 shown in Fig 4.7 can be activated in the previous 2 time steps before dispense operation to make dispensing operation instantaneous.

Detection time: Depending on the type of assay we may need different types of peripheral operations in the detect step, for e.g. heater, photodiode etc. The time taken for detect operation is assay specific. We assume that each detect operation is a unit step operation. The time taken for detection can be added later according to the type of detect operation.

4.4 Implementation details

The routing based synthesis (RBS) simulator developed in this work is modular, general purpose and can be used to simulate any synthesis algorithm. It is programmed in C++ using the object oriented paradigm. We use the same input file format as used by the DSSS simulator [9], i.e., a DMFB architecture file and an assay description file. The DMFB architecture file contains the dimensions and the location of input output ports and peripherals. The assay is specified as a directed acyclic graph, where each node represents the operations to be performed and the edges represent dependencies between the nodes.

The simulator models every part of the entire simulation process such that it makes the code modular and easy to change. We divide the simulation process into distinct categories and define the interactions between them. We model the hardware, i.e., the DMFB array on which we execute our assays and the droplets. The DMFB array is an array of electrodes. As shown in the Fig 4.10, we model each electrode as an object with properties like a unique ID, its location when used in the array, its type and whether it is occupied or not. We use this electrode object to model the DMFB array object. We also model each IO ports on the DMFB array with properties like ID, name, location, fluid dispensed, dispensing time etc. The DMFB array has properties like architecture name, its dimensions, an array of electrode objects, a list of input output port objects and the time step for each operation.

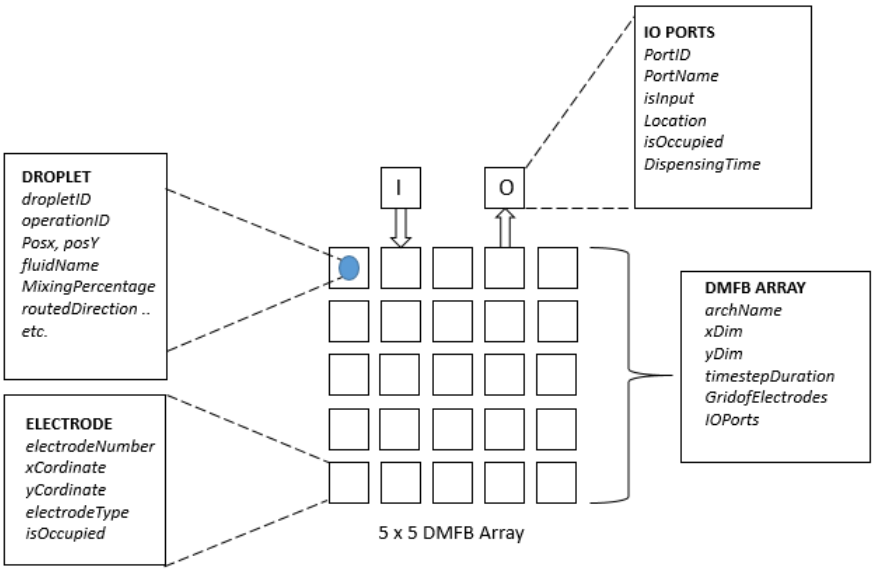

Figure 4.10: Hardware data structure

We read in the DMFB architecture file and instantiate the objects described above accordingly. We also model a droplet which is another major building block for the simulator. We model the droplet such it has its unique ID and the operation ID/node it is associated with. We also keep track of its current position, routing path, mix percentage if it is a mix droplet, its current status (active or inactive) and whether it moved in the current time step. To construct the assay DAG, we first model each node such that it includes information such as its ID, operation type, its parent nodes, children nodes, execution time stamps etc. We then build the DAG based on the assay file read in by the simulator.

We also built an interface to all standard operations performed on the DMFB array. We built routines such as dispensing droplet which instantiates a new droplet object on the DMFB object. We also have standard functions which perform operations like moving a droplet etc. By using these routines the simulator can be extended to simulate any synthesis algorithm.

4.5 Results

Experiment 1: Comparison of assay completion time with GPFP architecture

To evaluate the performance of RBS, we first compare the assay completion time with the List scheduling algorithm used on a GPFP architecture defined in [8]. The assay completion time on the GPFP array is compared with a DMFB array with direct addressing. The results in [8] show that the assay completion on the GPFP array is comparable to the DA array when the number of electrodes on DA array are approximately 3 times more than that of the GPFP array. To evaluate performance of the RBS algorithm, we first compare its assay completion time with the time required by list scheduling for the same assay on the GPFP architecture using an array of same dimensions. Because of the modular nature of the GPFP architecture, the number of electrodes on the DA array is approximately 26% more than the electrodes on the GPFP array. We evaluate the RBS algorithm using the same set of benchmarks used in the previous section, namely the PCR assay, in-vitro assay and protein split assays. We modify the scheduling algorithm for GPFP such that the assumptions made for RBS match with the GPFP scheduling algorithm i.e., dispense and detect are both considered as unit time operations. The results of the comparison is summarized in Table 4.1.

Table 4.1: Comparison of RBS with GPFP

Assay	Array Dimensions	GPFP Electrodes	RBS Electrodes	% Increase in Electrodes	Assay completion time		
					GPFP	RBS	% Improvement
B1_PCR	13×12	113	156	27.56	6630	4260	35.75
B1_PCR	13×15	145	195	25.64	6630	4190	36.79
B2_InVitro_Ex1_2s_2r	13×9	87	117	25.64	2300	1670	27.39
B2_InVitro_Ex1_2s_2r	13×12	113	156	27.56	2300	1600	30.43
B2_InVitro_Ex2_2s_3r	13×9	87	117	25.64	4470	1900	57.49
B2_InVitro_Ex2_2s_3r	13×12	113	156	27.56	2300	1580	31.3
B2_InVitro_Ex3_3s_3r	13×9	87	117	25.64	6660	2020	69.67
B2_InVitro_Ex3_3s_3r	13×12	113	156	27.56	4470	1860	58.39
B2_InVitro_Ex4_3s_4r	13×9	87	117	25.64	6700	2520	62.39
B2_InVitro_Ex4_3s_4r	13×12	113	156	27.56	4470	2150	51.9
B2_InVitro_Ex5_4s_4r	13×9	87	117	25.64	8870	4200	52.65
B2_InVitro_Ex5_4s_4r	13×12	113	156	27.56	6680	2460	63.17
B4_ProteinSplit_1_Eq	13×9	87	117	25.64	13240	8720	34.14
B4_ProteinSplit_1_Eq	13×12	113	156	27.56	13240	8570	35.27
B4_ProteinSplit_2_Eq	13×12	113	156	27.56	15460	10320	33.25
B4_ProteinSplit_2_Eq	13×15	145	195	25.64	15460	10260	33.64
B4_ProteinSplit_3_Eq	13×12	113	156	27.56	28670	12930	54.9
B4_ProteinSplit_3_Eq	13×15	145	195	25.64	17700	12490	29.44
B4_ProteinSplit_4_Eq	13×18	174	234	25.64	41880	15710	62.49
B4_ProteinSplit_4_Eq	13×21	203	273	25.64	30910	28580	7.54

Observations

Based on the results summarized in Table 4.1, we observe that RBS consistently shows an average reduction of 43% in the assay completion time for benchmarks used to evaluate it. This can be attributed to two key factors: improvement in the mixing time and increased parallelism during assay execution.

The time taken for a mixing operation to complete on a 2×5 module on the GPFP array is 2200 milliseconds. This mixing time reduces by 32% on average by using the routing-based

approach for mixing over the traditional module-based approach. This can be observed in assays like B1_PCR, B2_InVitro_Ex1_2s_2r in which the number of operations that can be executed in parallel are less than or equal to the number of available modules. In this case the improvement we see in the assay completion time can be attributed to the better mixing time, because of the routing-based mixing approach.

The inherent limitation of the GPFP architecture is an upper bound on the number of operations that can be executed in parallel. For example the 13×9 GPFP architecture has four modules and therefore we cannot execute more than four operations in parallel. However, larger assays usually have many operations that can be executed in parallel. For example B2_InVitro_Ex2_2s_3r has six operations that can be executed in parallel. When scheduled on a four module 13×9 GPFP array, the operations have to wait until the resources are available thereby increasing the execution time significantly. Comparing this with RBS synthesis we observe an average improvement of 57% which can be attributed to both the improved mixing quality and better parallelism. Since RBS has no limitation on the number of operations, we can schedule on the chip as opposed to virtual modules on the GPFP array, so RBS offers much better parallelism. When the same assay is scheduled on a larger 6 module 13×12 GPFP array we observe that the assay is completed in half the time taken by the assay on a 13×9 GPFP array. The only improvement we see in this case is due to better mixing time.

Based on this experiment, we can conclude that RBS offers reduction in assay completion time by approximately 30% due to better mixing time and another 20% - 25% improvement by executing maximum possible operations in parallel. Since we observe a better assay completion time than the GPFP architecture, we can conclude that RBS offers significant improvement over the traditional synthesis approach involving binding, scheduling, placement and routing. With RBS we simplify the synthesis problem into an easier routing problem in contrast to the traditional approach. Therefore the execution time of RBS is much smaller compared to approaches specified in [27], [7], [29] etc.

Experiment 2: Comparison of assay execution time with GRASP [37]

In [37], the author proposes a routing based-synthesis algorithm GRASP for DMFBs. GRASP stands for greedy randomized adaptive search procedure. This algorithm finds the routes for operation completion using a randomized and greedy approach. At each time step, the algorithm creates a list of possible directions the droplet can move in and randomly selects one out of them. To evaluate the performance of RBS against GRASP we use the same set of synthetic benchmarks used in [37]. These benchmarks include a random sequence of operations that can be performed on any microfluidic array. The name of each benchmark specifies the number of operations in it. We compare the assay completion time for the given benchmarks using arrays of different dimensions. The results of this comparison are summarized in Table 4.2

Table 4.2: GRASP vs RBS results

Benchmark	Size	GRASP	RBS	%age	Size	GRASP	RBS	%age	Size	GRASP	RBS	%age
Graph 10	6×6	4777	3689	22.78	10×10	4119	3537	14.13	15×15	4034	3499	13.26
Graph 20	6×6	7574	4296	43.28	10×10	5019	3996	20.38	15×15	4809	4079	15.18
Graph 30	6×6	11401	5779	49.31	10×10	6797	4629	31.90	15×15	6478	4416	31.83
Graph 40	6×6	16670	5678	65.94	10×10	6317	4422	30.00	15×15	6101	4285	29.77
Graph 50	8×8	11887	8011	32.61	10×10	9712	7542	22.34	15×15	8853	7936	10.36
Graph 60	6×6	11750	8603	26.78	10×10	10262	8281	19.30	15×15	9639	8639	10.37
Graph 70	8×8	29587	19894	32.76	10×10	21049	15116	28.19	15×15	17897	14851	17.02
Graph 80	8×8	19622	15392	21.56	10×10	15546	12515	19.50	15×15	12970	11084	14.54

Observations

With RBS, we have an average 26% improvement in assay execution time across all assays and all dimensions. GRASP relies on a randomized approach in determining the next step the droplet should take to complete the mixing operation. This may or may not be the best step the droplet can take at that instant. For example in benchmark Graph10 executing on a 6×6 array, we observe that the mixing operations using RBS is 65% better on average when compared to GRASP. We observe that RBS performs better in the case of smaller array

dimensions e.g. 6×6 or 8×8, with an average 35% improvement in the assay completion time. This can be attributed to the congestion on the chip due to same number of droplets on a smaller arrays. With GRASP, droplets frequently end up moving in 180°, i.e., opposite to the current direction, which results in negative mixing. In RBS, we avoid moving the droplet in 180° unless the droplet reaches a deadlock state.

GRASP is a randomized algorithm, therefore to obtain the best results we have to let the algorithm run multiple times for it to explore the search space and determine the best solution. RBS on the other hand is a deterministic algorithm which greedily selects the best possible move at any instant. Therefore, RBS is much more suited for field programmable operations which require fast synthesis.

Chapter 5

Conclusion and Future Work

The GPFP architecture proposed in [4] overcomes the limitations of direct addressing DMFBs by reducing the pin count. It also simplifies the synthesis problem to a scheduling, binding and routing problem which facilitates fast synthesis, thereby making it truly field programmable. However, the limitation of the GPFP architecture is the fixed number of modules. For example a 13×9 DMFB array has 4 modules, so it can execute a maximum of 4 operations in parallel. If any of the electrodes forming modules on the GPFP array is faulty, the module is rendered useless for operation execution, thereby reducing the number of available modules. Depending on the number of parallel operations in an assay executed on this faulty chip, the assay execution time might increase by as much as 100 %. In this thesis we make these modules fault-tolerant by reconfiguring them based on the location and type of faults. We reconfigure faulty 2×5 modules in the GPFP array into smaller mixers which can complete the mixing operation in comparable time. We use an analytical approach to determine the percentage of mixing completed when the droplet moves in any direction. The results show that reconfiguration of faulty modules can improve the assay completion time by 23% on an average, when compared to execution time of the same assay on an array with a faulty module.

We then extend this analytical approach to implement a routing based synthesis (RBS)

methodology for DMFB arrays. Since mixing can be performed by routing a droplet on any random sequence of electrodes, we can modify the synthesis problem into a droplet routing problem. By transforming the synthesis problem into a routing problem we reduce its computational complexity drastically. This routing based sythesis approach can now be used to perform in-field fast synthesis. This approach can be used to synthesize the assay even in the presence of faulty electrodes. To evaluate RBS, we compare its results with the assay completion time on the GPFP architecture. We observe that RBS offers 43% improvement in the assay completion time. We also compare RBS with GRASP, a greedy randomized routing based synthesis algorithm. RBS offers approximately 26% improvement over GRASP owing to better mixing and merging operations. The execution time of GRASP is much higher due to its randomized nature.

RBS is a greedy algorithm which selects the next step based on the current position of each droplet. It does not consider collision or deadlock situations that may arise by moving droplets without considering the location of other droplets. With the information about location of other droplets, we can route droplets such that they are evenly distributed on the electrode array. We can also reduce the negative movements or waiting at the current location, if we route droplets by considering other droplets. Contamination occurs when droplets get adsorbed on the surface of the electrode, thereby contaminating other droplets that pass over the surface of this contaminated electrode. To avoid contamination, we enclose the droplet in a filler medium like silicone oil. An alternative approach would be to use wash droplets. RBS can be modified to include wash droplets to clean the surface of the contaminated electrodes effectively.

Bibliography

[1] Arjan Floris, Steven Staal, Stefan Lenk, Erik Staijen, Dietrich Kohlheyer, Jan Eijkel, and Albert van den Berg. A prefilled, ready-to-use electrophoresis based lab-on-a-chip device for monitoring lithium in blood. *Lab on a Chip*, 10(14):1799–1806, 2010.

[2] Albert van den Berg. Labs on a chip for health care applications. In *The 14th International Conference on Miniaturized Systems for Chemistry and Life Sciences*, 2010.

[3] Daniel P Rose, M Ratterman, Daniel K Griffin, Liwen Hou, Nancy Kelley-Loughnane, Rajesh Naik, Joshua Hagen, Ian Papautsky, Jason Heikenfeld, et al. Adhesive rfid sensor patch for monitoring of sweat electrolytes. 2014.

[4] Hongying Zhu, Sam Mavandadi, Ahmet F Coskun, Oguzhan Yaglidere, and Aydogan Ozcan. Optofluidic fluorescent imaging cytometry on a cell phone. *Analytical Chemistry*, 83(17):6641–6647, 2011.

[5] Biochips Global Market iq4i research and consultancy report. `$http://www.pr.com/press-release/619088$`. Accessed: 2015-10-02.

[6] Fei Su, Sule Ozev, and Krishnendu Chakrabarty. Testing of droplet-based microelectrofluidic systems. volume 0, page 1192. IEEE, 2003.

[7] Fei Su and Krishnendu Chakrabarty. Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In *Proceedings of the 42nd annual Design Automation Conference*, pages 825–830. ACM, 2005.

[8] Rissen Alfonso Joseph. *A General Purpose Field-Programmable Digital Microfluidic Biochip with Scannable Electrofluidic Control.* PhD thesis, University of Cincinnati, 2014.

[9] Daniel Grissom and Philip Brisk. A field-programmable pin-constrained digital microfluidic biochip. In *Proceedings of the 50th Annual Design Automation Conference*, page 46. ACM, 2013.

[10] Madhuri N Gupta. *Multi-Board Digital Microfluidic Biochip Synthesis with Droplet Crossover Optimization.* PhD thesis, University of Cincinnati, 2014.

[11] Elena Maftei, Paul Pop, and Jan Madsen. *Synthesis of digital microfluidic biochips with reconfigurable operation execution.* PhD thesis, Technical University of Denmark-Danmarks Tekniske Universitet, Department of Applied Mathematics and Computer ScienceInstitut for Matematik og Computer Science, Software EngineeringSoftware Engineering, 2011.

[12] David Juncker, Heinz Schmid, Ute Drechsler, Heiko Wolf, Marc Wolf, Bruno Michel, Nico de Rooij, and Emmanuel Delamarche. Autonomous microfluidic capillary system. *Analytical chemistry*, 74(24):6139–6144, 2002.

[13] Anthony K Au, Hoyin Lai, Ben R Utela, and Albert Folch. Microvalves and micropumps for biomems. *Micromachines*, 2(2):179–220, 2011.

[14] Howard A Stone, Abraham D Stroock, and Armand Ajdari. Engineering flows in small devices: microfluidics toward a lab-on-a-chip. *Annu. Rev. Fluid Mech.*, 36:381–411, 2004.

[15] Venkatachalam Chokkalingam, Jurjen Tel, Florian Wimmers, Xin Liu, Sergey Semenov, Julian Thiele, Carl G Figdor, and Wilhelm TS Huck. Probing cellular heterogeneity in cytokine-secreting immune cells using droplet-based microfluidics. *Lab on a Chip*, 13(24):4740–4744, 2013.

[16] Yung-Chieh Tan, Vittorio Cristini, and Abraham P Lee. Monodispersed microfluidic droplet generation by shear focusing microfluidic device. *Sensors and Actuators B: Chemical*, 114(1):350–356, 2006.

[17] Piotr Garstecki, Michael J Fuerstman, Howard A Stone, and George M Whitesides. Formation of droplets and bubbles in a microfluidic t-junctionscaling and mechanism of break-up. *Lab on a Chip*, 6(3):437–446, 2006.

[18] Krishnendu Chakrabarty and Fei Su. *Digital microfluidic biochips: synthesis, testing, and reconfiguration techniques*. CRC Press, 2006.

[19] Michael George Pollack. *Electrowetting-based microactuation of droplets for digital microfluidics*. PhD thesis, Duke University, 2001.

[20] Vijay Srinivasan, Vamsee K Pamula, and Richard B Fair. An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids. *Lab on a Chip*, 4(4):310–315, 2004.

[21] William L Hwang, Fei Su, and Krishnendu Chakrabarty. Automated design of pin-constrained digital microfluidic arrays for lab-on-a-chip applications*. In *Proceedings of the 43rd annual Design Automation Conference*, pages 925–930. ACM, 2006.

[22] Tao Xu and Krishnendu Chakrabarty. Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips. In *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 112–117. ACM, 2006.

[23] Tao Xu and Krishnendu Chakrabarty. Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips. In *Proceedings of the 45th annual Design Automation Conference*, pages 173–178. ACM, 2008.

[24] Shih-Kang Fan, Craig Hashi, and Chang-Jin Kim. Manipulation of multiple droplets on n× m grid by cross-reference ewod driving scheme and pressure-contact packaging. In *Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE The Sixteenth Annual International Conference on*, pages 694–697. IEEE, 2003.

[25] Joo Hyon Noh, Jiyong Noh, Eric Kreit, Jason Heikenfeld, and Philip D Rack. Toward active-matrix lab-on-a-chip: programmable electrofluidic control enabled by arrayed oxide thin film transistors. *Lab on a Chip*, 12(2):353–360, 2012.

[26] Vaishnavi Ananthanarayanan and William Thies. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of biological engineering*, 4(1):1–13, 2010.

[27] Fei Su and Krishnendu Chakrabarty. Architectural-level synthesis of digital microfluidics-based biochips. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 223–228. IEEE Computer Society, 2004.

[28] Ping-Hung Yuh, Chia-Lin Yang, and Yao-Wen Chang. Placement of defect-tolerant digital microfluidic biochips using the t-tree formulation. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3(3):13, 2007.

[29] Tao Xu and Krishnendu Chakrabarty. Integrated droplet routing and defect tolerance in the synthesis of digital microfluidic biochips. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(3):11, 2008.

[30] Vijay Srinivasan, Vamsee Pamula, Michael Pollack, and Richard Fair. A digital microfluidic biosensor for multianalyte detection. In *Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE The Sixteenth Annual International Conference on*, pages 327–330. IEEE, 2003.

[31] Fei Su and Krishnendu Chakrabarty. Module placement for fault-tolerant microfluidics-

based biochips. In *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, volume 11, pages 682–710. ACM, 2004.

[32] Fei Su and Krishnendu Chakrabarty. Defect tolerance for gracefully-degradable microfluidics-based biochips. In *VLSI Test Symposium, 2005. Proceedings. 23rd IEEE*, pages 321–326. IEEE, 2005.

[33] Mirela Alistar, Paul Pop, and Jan Madsen. *Compilation and Synthesis for Fault-Tolerant Digital Microfluidic Biochips*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Informatics and Mathematical ModelingInstitut for Informatik og Matematisk Modellering.

[34] Phil Paik, Vamsee K Pamula, and Richard B Fair. Rapid droplet mixers for digital microfluidic systems. *Lab on a Chip*, 3(4):253–259, 2003.

[35] MG Pollack, AD Shenderov, and RB Fair. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip*, 2(2):96–101, 2002.

[36] Daniel Grissom, Kenneth O'Neal, Benjamin Preciado, Hiral Patel, Robert Doherty, Nick Liao, and Philip Brisk. A digital microfluidic biochip synthesis framework. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 177–182. IEEE, 2012.

[37] Elena Maftei, Paul Pop, and Jan Madsen. Routing-based synthesis of digital microfluidic biochips. *Design Automation for Embedded Systems*, 16(1):19–44, 2012.