Synthesis of Reliable Digital Microfluidic Biochips using Monte Carlo Simulation

Elena Maftei[1], Paul Pop[1], Florin Popenţiu Vlădicescu[2]
[1]*Dept. of Informatics and Mathematical Modelling,*
Technical Univ. of Denmark, DK-2800 Kgs. Lyngby, Denmark
em@imm.dtu.dk, paul.pop@imm.dtu.dk
[2]*Faculty of Electrical Engineering and Information Technology,*
Univ. of Oradea, RO-410087 Oradea, Romania
popentiu@imm.dtu.dk

ABSTRACT: Microfluidic-based biochips are replacing the conventional biochemical analyzers, and are able to integrate on-chip all the necessary functions for biochemical analysis using microfluidics. The "digital microfluidic" biochips are based on the manipulation of liquids not as a continuous flow, but as discrete droplets (hence the term "digital"), and thus are highly reconfigurable and scalable. We model a biochemical application using an abstract model consisting of a sequencing graph. The digital biochip is modeled as a two-dimensional array of cells, where each cell can hold a droplet. In this paper we propose an integer linear programming (ILP) synthesis methodology that, starting from a biochemical application and a given biochip, determines the allocation, placement, resource binding, and scheduling of the operations in the application. Our goal is to find that particular implementation of an application onto a biochip, which has the highest probability to be reconfigured successfully in case of multiple faulty cells. We propose a fault model for biochips, and use Monte Carlo simulation to evaluate the probability of successful reconfiguration of each implementation in case of faults. The proposed methodology has been evaluated using a real-life example.

1 INTRODUCTION

Microfluidic-based biochips (also referred to as lab-on-a-chip) are replacing the conventional biochemical analyzers, and are able to integrate on-chip all the necessary functions for biochemical analysis using microfluidics, such as, transport, splitting, merging, dispensing, mixing, and detection.

Applications areas of biochips include: clinical diagnostics, bio-defense applications, massively parallel DNA analysis and automated drug discovery (Thorsen et al. 2002). Biochips are able to: provide miniaturization, thus enabling very small volumes and speeding up chemical reactions and analytical detection; obtain higher throughput with minimal human intervention; use smaller sample and reagent consumption; provide higher sensitivity at significantly lower costs per assay than the traditional methods; and increase productivity through automation and parallelization (Thorsen et al. 2002).

There are two approaches to microfluidics. The "first generation" is based on the continuous flow of liquid through micro-channels using micropumps and microvalves (Verpoorte and Rooij 2003) The second approach, also called "second generation", is based on the manipulation of liquids not as a continuous flow, but as discrete droplets (Pollack et al. 2002). Thus, the second type of microfluidic biochips is also referred to as "digital microfluidics", due to the analogy between the droplets and the bits in a digital system.

Although the continuous-flow biochips have been used for simple biochemical applications, due to their lack of flexibility they are unsuitable for more complex applications that require complicated fluid manipulations (Zhang et al. 2002). Therefore, in this paper, we are interested in droplet-based digital biochips, which are highly reconfigurable and scalable.

1.1 Related Work

CAD tools for digital microfluidics are in their infancy, and designers are using manual, bottom-up, full-custom, design approaches to implement such biochips (Chakrabarty and Zeng 2005). However, digital microfluidic biochips are becoming increasingly complex, and are expected to be integrated with microelectronic components in next generation system-on-chips. Consequently, the current bottom-up full-custom design approach will not scale to the new designs. Therefore, new top-down methods and techniques are required, which can offer the same level of support as the one taken for granted currently in the semiconductors industry. Such techniques will reduce the design cost and improve productivity, and are the key to the further growth and market penetration of

biochips (Chakrabarty and Zeng 2005).

Considering their architecture and the design tasks that have to be performed, the design of digital microfluidic biochips has similarities to high level synthesis of VLSI systems (Gajski et al. 1992; Micheli 1994). Motivated by this similarity, a few researchers have recently started to propose approaches for the top-down design of such biochips. The following are the main design tasks that have been addressed:

- During the design of a digital microfluidic biochip, the bioassay protocols have to be mapped to the on-chip modules. The protocols are *modeled* using process graph models (Chakrabarty and Su 2006), where each node is an operation, and each edge represents a dependency.
- Once the protocol has been specified, the necessary modules for the implementation of the protocol operations will be selected from a module library (Su and Chakrabarty 2004). This is called the *allocation* step.
- As soon as the *binding* of operations to the allocated modules is decided (Su and Chakrabarty 2004), the *scheduling* (Su and Chakrabarty 2004; Ricketts et al. 2006) step determines the time duration for each bioassay operation, subject to resource constraints and precedence constraints imposed by the protocol.
- Finally, chip will be synthesized according to the constraints on the types of resources, cost, area and protocol completion times. During the chip synthesis, the *placement* (Su and Chakrabarty 2006) of each module on the microfluidic array and the *routing* (Su et al. 2006; Cho and Pan 2008) of droplets from one module to another have to be determined.
- All of the presented design tasks have to take into account possible defects during the fabrication of the microfluidic biochip. Thus, *testing* (Xu and Chakrabarty 2007; Kerkhoff 2007) and *reconfiguration* (Su and Chakrabarty 2006) have to be performed.

In this paper we propose an integer linear programming (ILP) synthesis methodology that, starting from a biochemical application modeled as a sequencing graph and a given biochip, determines the allocation, placement, resource binding, and scheduling of the operations in the application. Such a digital microfluidic biochip is a dynamically reconfigurable system. We have extended the model from (Su and Chakrabarty 2004), which considers a given allocation and proposes an ILP model only for scheduling and binding (i.e., without considering allocation and placement), and without taking into account faulty cells.

If multiple cells become faulty, the microfluidic operation can be moved to another part of the array by changing the control voltages applied on the electrodes. Our goal is to find that particular implementation of an application onto an array, which has the highest probability to be reconfigured successfully in case of multiple faults. We propose a fault model for biochips, and use Monte Carlo simulation to evaluate the probability of successful reconfiguration of each implementation in case of faults.

The paper is organized in six sections. Sections 2.1 and 2.2 present the model of the digital microfluidic biochip and the fault model, respectively. We introduce the sequencing graph model we use to capture a biochemical application in Section 2.3. We formulate the problem in Section 3 and illustrate the design tasks using several examples. The proposed ILP model and the Monte Carlo simulation approach used are presented in Section 4. The evaluation of the proposed approach is performed in Section 5. The last section presents our conclusions.

2 SYSTEM MODEL

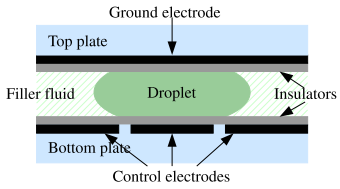
2.1 *Digital Microfluidic Biochip Architecture*

In a digital microfluidic biochip the manipulation of liquids is performed using discrete droplets. There are several mechanisms for droplet manipulation (Fair 2007). Our proposed research will consider electrowetting-on-dielectric (EWD) (Pollack et al. 2002), but can be extended to handle other techniques as well. EWD is the most promising technique, and can provide high droplet speeds of up to 20 cm/s.

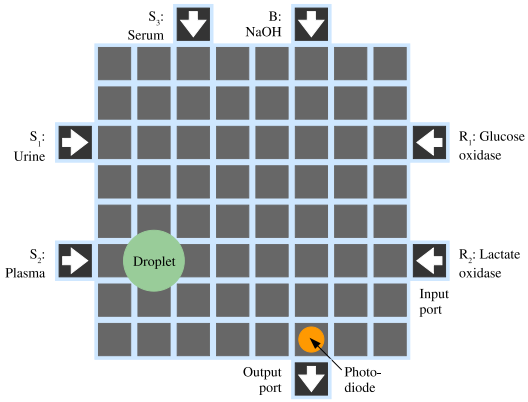
A biochip is composed of several cells, see Figure 1(b). Let us discuss first how a cell is functioning, and then we will show how cells are put together to form a chip.

The schematic of a cell is presented in Figure 1(a). The droplet is sandwiched between two glass plates (the top plate and the bottom plate), and moves within a filler fluid. The top plate contains a single ground electrode, while the bottom plate has several control electrodes. The electrodes are insulated from the droplet trough an insulation material. Considering an EWD approach, the movement of droplets is controlled by applying voltages to the required electrodes. For example, turning off the middle control electrode and turning on the right control electrode in Figure 1(a) will force the droplet to move to the right. For the details of the EWD-based chip fabrication, the reader is directed to (Pollack et al. 2002).

Several cells are put together to form a two-dimensional array (an example architecture is presented in Figure 1(b)). Using EWD manipulation, droplets can be moved to any location without the need for pumps and valves, which are required in a continuous-flow biochip. Besides the basic cell dis-



(a) Cell architecture



(b) Biochip: array of cells

Operation	Area (cells)	Time (s)
Mixing	2x2	6
Mixing	2x3	5
Mixing	2x4	4
Dilution	2x2	6
Dilution	2x3	5
Dilution	2x4	4
Detection	1x1	10
Storage	1x1	–

(c) Module library

Figure 1: Biochip architecture

cussed previously, the chip typically contains input and output ports and detectors. The detection can be done by using, for example, a LED beneath the bottom plate and a photodiode on the top plate.

Using this architecture, and changing correspondingly the control voltages, several operations, such as transport, splitting, merging, dispensing, mixing, and detection, can be performed. For example, mixing is done by transporting two droplets to the same location, and then moving them next to each other on a circular path within a delimited cell block. Any cells in the chip can be used for such an operation, thus, we say that the chip is "reconfigurable". This property is particulary useful in case certain cells are faulty, because the operation can be simply moved to another part of the chip.

As is the case with digital circuits, we consider that designers will build and characterize a module library $\mathcal{L}$, where for each operation there are several options varying in terms of area and execution time, see Figure 1(c).

2.2 *Fault Model*

The types of faults for a microelectronic chip are well known. However, biochips belong to the class of Micro-Electro-Mechanical Systems (MEMS), and thus exhibit different types of faults (Deb and Blanton 2000).

In this paper we will concentrate on permanent faults, where a cell is simply no longer capable of droplet manipulation, as opposed to parametric faults, which cause deviations in the system performance. Let us illustrate on type of permanent fault using Figure 1(a), where we depict the architecture of a cell. Suppose there is a short between two adjacent electrodes. In this case, the two electrodes will practically form one larger electrode. When a droplet is moved to this electrode, it is no longer large enough to cover the inter-electrode gap, and hence further movement to another cell is not possible.

In this paper, we will not present all the possible faults. For an in-depth discussion, the reader is directed to (Xu and Chakrabarty 2007). A summary of the types of permanent faults and corresponding errors that can affect digital microfluidic biochips is presented in Table 1, using the terminology from (Avizienis et al. 2001).

Table 1: Fault types (adapted from Xu 2007)

Cause	Fault	Fault Model	Error
Excessive voltage applied to electrode	Dielectric breakdown	Droplet-electrode short (a short between the droplet and the electrode)	Droplet undergoes electrolysis, which prevents its further transportation
Electrode actuation for excessive duration	Irreversible charge concentration on an electrode	Electrode-stuck-on (the electrode remains constantly activated)	Unintentional droplet operations or stuck droplets
Excessive mechanical force applied to the chip	Misalignment of parallel plates (electrodes and ground plane)	Pressure gradient (net static pressure in some direction)	Droplet transportation without activation voltage
Coating failure	Non-uniform dielectric layer	Dielectric islands (islands of Teflon coating)	Fragmentation of droplets and their motion is prevented
Abnormal metal layer deposition and etch variation during fabrication	Grounding Failure	Floating droplets (droplets are not anchored)	Failure of droplet transportation
	Broken wire to control source	Electrode open (electrode actuation is not possible)	Failure to activate the electrode for droplet transportation
	Metal connection between two adjacent electrodes	Electrode short (short between electrodes)	A droplet resides in the middle of two shorted electrodes, and its transport along one or more directions cannot be achieved
Particle contamination or liquid residue	A particle that connects two adjacent electrodes	Electrode short	

Our fault model considers multiple faults. Microfluidic chips have not yet been manufactured in

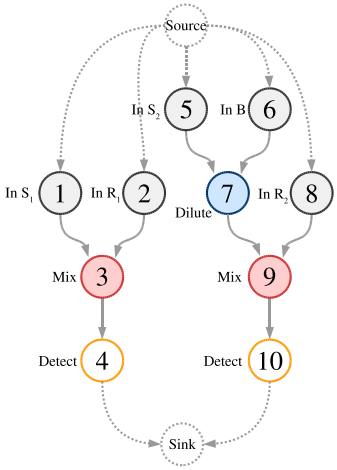


Figure 2: Example application

large quantities, and thus there are no statistics regarding failure probabilities. Since cells are identical, we think is reasonable to assume that each cell has the same failure probability, Q_{cell}. Our synthesis approach can use as an input any updated fault model, when the data becomes available.

In addition, we assume that, after fabrication, the faulty cells are detected with a technique such as the one described in (Xu and Chakrabarty 2007). If multiple cells are identified as faulty, the microfluidic operation can be moved to another part of the array by changing the control voltages applied on the electrodes, as described in Section 3.3.

2.3 *Biochemical Application Model*

We model a biochemical application using an abstract model consisting of a sequencing graph (Chakrabarty and Zeng 2005). The graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ is directed, acyclic and polar (i.e., there is a *source node*, which is a node that has no predecessors and a *sink node* that has no successors). Each node $O_i \in \mathcal{V}$ represents one operation. The binding of operations to modules in the architecture is captured by the function $\mathcal{B} : \mathcal{V} \rightarrow \mathcal{A}$, where $\mathcal{A} \subset \mathcal{L}$ is the set of allocated modules from the given library $\mathcal{L}$.

An edge $e_{i,j} \in \mathcal{E}$ from O_i to O_j indicates that the output of operation O_i is the input of O_j. An operation can be activated after all its inputs have arrived and it issues its outputs when it terminates. Operations are non-preemptable and thus cannot be interrupted during their execution.

We assume that, for each operation O_i, we know the execution time $C_i^{M_k}$ on module $M_k = \mathcal{B}(O_i)$ where it is assigned for execution. Currently, the routing time between two operations is an order of magnitude smaller compared to the operation time. Hence, we consider the routing time to be part of the operation execution time and do not model it explicitly.

3 PROBLEM FORMULATION

The problem we are addressing in this paper can be formulated as follows. Given (1) a biochemical application modeled as a graph $\mathcal{G}$, (2) a biochip consisting of a two-dimensional $m \times n$ array of cells, (3) a characterized module library $\mathcal{L}$ and (4) a time constraint $\delta_{\mathcal{G}}$ by which the application has to finish, we are interested to synthesize that implementation Ψ, which minimizes the completion time of the application (i.e., finishing time of the sink node, $t_{sink}^{finish} < \delta_{\mathcal{G}}$) and has the highest probability that it will be reconfigured successfully in case of multiple faulty cells.

Synthesizing an implementation $\Psi = < \mathcal{A}, \mathcal{P}, \mathcal{B}, \mathcal{S} >$ means deciding on: (1) the allocation $\mathcal{A} \subset \mathcal{L}$, which determines what modules from the library $\mathcal{L}$ should be used, (2) the placement $\mathcal{P}$ of the modules on the $m \times n$ array, (3) the binding $\mathcal{B}$ of each operation $O_i \in \mathcal{V}$ to a module $M_k \in \mathcal{A}$, and the schedule $\mathcal{S}$ of the operations, which contains the start time t_i^{start} of each operation O_i on its corresponding module.

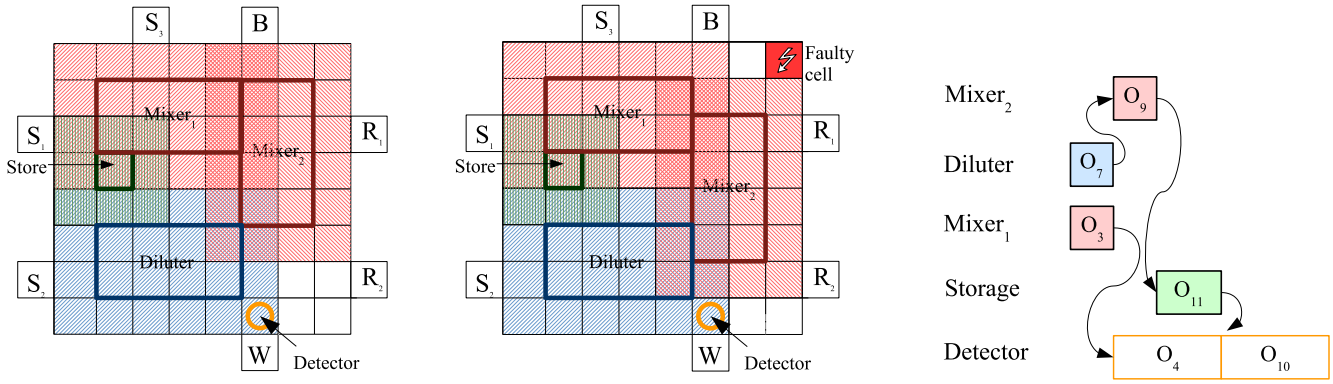
The next subsections will illustrate each of these subproblems.

3.1 *Allocation and placement*

Let us consider the application graph $\mathcal{G}$ in Figure 2, where we have ten operations, O_1 to O_{10}. We would like to implement this application on the 8x8 biochip from Figure 1(b). The input and detection operations are already assigned to the corresponding input ports and detection module, respectively. Thus, O_1 is assigned to the input port S_1, O_2 to R_1, O_5 to S_2, O_6 to B and O_8 to R_2. The detection operations O_4 and O_{10} will be performed by the on-chip detector, and then the droplet will be moved to the waste reservoir through the output port W. However, for the mixing operations (O_3 and O_9) and the dilution operation O_7 our synthesis approach will have to allocate the appropriate modules.

Let us assume that the available module library is the one captured by Figure 1(c). We have to select those modules that will lead to the minimum application completion time and place them on the 8x8 chip in such a way that, if multiple cells become faulty, there is a high chance that the placement of these modules can be changed to avoid the faulty cells. The optimal solution to the allocation and placement problem is presented in Figure 3(a), where the following modules are used: two 2x4 mixers ($Mixer_1$ and $Mixer_2$), one 2x4 diluter and one 1x1 "store" module.

Note that special "store" modules have to be allocated if a droplet has to wait before being processed. Consider the detector module. We have to perform two detection operations, O_4 and O_{10}. If the second droplet is routed to the detector before the first detection finishes, a 1x1 storage cell is required to store the droplet before it can be moved to the detector. In gen-



(a) No faults (b) Reconfiguration (c) Schedule

Figure 3: Implementation example

eral, if there exists an edge $e_{i,j}$ from O_i to O_j such that O_j is not immediately scheduled after O_i (i.e., there is a delay between the finishing time of O_i and the start time of O_j) then we will have to allocate a storage cell for $e_{i,j}$. Hence, the allocation of storage cells depends on how the schedule is constructed.

The placement for the discussed solution is as indicated in Figure 3(a), where we can notice that modules occupy a space larger than their size (the hashed area corresponding to each module). This is to avoid droplet-merging and contamination. If two droplets are next to each other on two adjacent cells, they will tend to merge to form one single droplet. Therefore, we consider for each module a border of one-cell size. For example, $Mixer_1$ which has a size of 2x4 will occupy 4x6 cells.

The main difference between our placement problem and the placement for microelectronic chips (Gajski et al. 1992; Micheli 1994) is that, in our case, modules can physically overlap on-chip as long as they do not overlap in time, i.e., they are used during different time intervals. This property is due to the reconfigurability of the digital microfluidic biochip. After an operation has finished executing on a module, we can reuse the same cells as part of another module.

3.2 *Binding and Scheduling*

Once the modules have been allocated and placed on the cell array, we have to decide where to execute the operations (binding) and in which order (scheduling), such that the application completion time is minimized.

Considering the graph in Figure 2 and the modules in Figure 3(a), Figure 3(c) presents the optimal schedule. The schedule is depicted as a Gantt chart, where, for each module, we represent the operations as rectangles with their length corresponding to the duration of that operation on the module. For example, operation O_9 is bound to module $Mixer_2$ (i.e., $\mathcal{B}(O_9) = Mixer_2$). O_9 starts immediately after the dilution operation O_7 (i.e, $t_9^{start} = 4$) and takes 4 s, finishing at time $t_9^{finish} = 8$ s. The total schedule length will be 24 s. We consider that the schedule is divided in time-steps of one second, and we capture the set of time-steps with $\mathcal{T}$.

Note that a new operation has been introduced, O_{11}, which corresponds to the storing of the second droplet before undergoing detection.

3.3 *Reconfiguration in Case of Faults*

In this paper we are interested in that implementation Ψ which, not only minimizes the schedule length, but also has a high chance to be reconfigured successfully in case of faults. For a given fault scenario, we denote the set of faults with $\mathcal{F}$. Let us assume that there is a fault in cell number 8 (counted from the top-left corner) of the biochip, as depicted with a lightning symbol in Figure 3(b), i.e., $\mathcal{F} = \{c_8\}$.

In this case, we would have to reconfigure the chip such that it does not use the faulty cell. The affected module in this case is $Mixer_2$, and it can be reconfigured as presented in the Figure 3(b).

4 ILP-BASED SYNTHESIS

The problem presented in the previous section is NP-complete (scheduling in even simpler contexts is NP-complete (Ullman 1975)). Our general strategy is to split this problem into two steps:

1. In the first step we generate several solutions Ψ_i for different area constraints $m \times n$ and time constraints $\delta_{\mathcal{G}}$ imposed by the designer. Each solution has the minimum schedule length for the imposed area constraints, but will have different allocation, binding and placement of modules. We have developed an ILP model, which is presented in Section 4. Using this model we use an ILP solver to obtain those implementations that minimize the schedule length under the imposed constraints. Let us call this step ILP/S.

2. Given an implementation Ψ_i, we evaluate its successful reconfiguration probability as follows. We generate several faulty cells using MCS. For each fault scenario $\mathcal{F}$, we attempt to reconfigure the chip such that it will not use these. We do not change the allocation in the implementation Ψ_i under evaluation (i.e., the same modules have to be used), but we allow the changing of allocation, binding, placement and schedule, under the constraint that the schedule length of Ψ_i does not exceed the imposed timing constraint. This reconfiguration is also performed using the ILP solver, similar to the previously outlined synthesis step. We name this reconfiguration step ILP/R.

4.1 *ILP Model*

In this section an integer linear programming (ILP) approach for solving the problem is presented. Thus, a system is described by a minimization objective and a set of constraints which define valid conditions for the system variables. A solution to the modeled problem is an enumeration of all system variables, such that the constraints are satisfied.

The optimization objective is specified as minimizing the completion time of the application,

$$\text{minimize } t_{sink}^{finish}, \tag{1}$$

where t_{sink}^{finish} is the finishing time of the sink node of the application.

The constraints fall under the following categories: (i) scheduling and precedence, (ii) resource, (iii) placement and (iv) fault-tolerance constraints. In order to be able to express them, a binary variable is defined as follows:

$$z_{i,j,k,l} = \begin{cases} 1, & \text{if operation } O_i \text{ starts executing at} \\ & \text{time-step } j \text{ on module } M_k \text{ placed} \\ & \text{with its top-left corner over cell } c_l \\ 0, & \text{otherwise} \end{cases}$$

Such a variable captures the allocation and binding (operation O_i is executing on module M_k), the scheduling (O_i starts to execute at time-step j, with a duration of $C_i^{M_k}$) and the placement (the top-left corner of module M_k is placed over cell c_l). For example, considering the dilution operation implemented as in Figure 3(a) the binary variable will be expressed as:

$$z_{i,j,k,l} = \begin{cases} 1, & \text{if i=7, j=1, k=Diluter, l=33} \\ 0, & \text{otherwise} \end{cases}$$

By using the defined variable, the start time of an operation $O_i \in \mathcal{V}$ becomes:

$$t_i^{start} = \sum_j \sum_k \sum_l j \times z_{i,j,k,l}, \quad \forall O_i \in \mathcal{V}, \tag{2}$$

where j represents the time-step when the operation starts executing.

4.1.1 Scheduling and precedence constraints

The scheduling constraint requires that every operation O_i be scheduled only once:

$$\sum_j \sum_k \sum_l z_{i,j,k,l} = 1, \quad \forall O_i \in \mathcal{V}. \tag{3}$$

For each edge in the application graph we have to introduce a precedence constraint. Consider the operations O_i and $O_n \in \mathcal{V}$ for which there exists a dependency $e_{i,n} \in \mathcal{E}$ in the sequencing graph $\mathcal{G}$. Then O_n must be scheduled for execution only after the completion of O_i:

$$t_i^{start} + \sum_j \sum_k \sum_l \left(C_i^{M_k} \times z_{i,j,k,l} \right) \leq t_n^{start}, \tag{4}$$

$$\forall O_i \text{ and } O_n \text{ such that } \exists e_{i,n} \in \mathcal{E}.$$

For example, considering operations O_9 and O_{10} in Figure 2, with O_{10} depending on O_9, we have $t_9^{finish} \leq t_{10}^{start}$. If O_n is not scheduled immediately after the completion of O_i then a storage module is required. The number of such storage modules during a time-step j is important in defining the placement constraints for the model, since the storage modules also occupy chip area. Using a binary variable $m_{i,j}$ defined as:

$$m_{i,j} = \begin{cases} 1, & \text{if a storage unit is needed for } O_i \text{ in step } j \\ 0, & \text{otherwise} \end{cases}$$

we can capture the number of storage units required during a time-step j. Thus, at time-step j, the binary variable associated with the edge between operations O_i and O_n is expressed as:

$$\sum_{h=1}^{j-C_i^{M_k}} \sum_k \sum_l z_{i,h,k,l} - \sum_{h=1}^{j} \sum_k \sum_l z_{n,h,k,l} = m_{i,j}, \tag{5}$$

$$\forall j \in \mathcal{T}, \forall O_i, O_n \in \mathcal{V} \text{ such that } \exists e_{i,n} \in \mathcal{E}$$

Variable $m_{i,j}$ will have the value 1 at that time-step j when O_i has finished executing (first sum of the equation equals 1), but O_j has not started yet (second term of the equation equals 0).

4.1.2 Resource constraints

Considering the fact that two operations of the same type can be bound to the same resource, a constraint must be expressed to prevent the overlapping of these operations during their execution. An operation O_i is executing at time-step j if:

$$\sum_{h=j-C_i^{M_k}+1}^{j} \sum_k \sum_l z_{i,h,k,l} = 1, \quad \forall O_i \in \mathcal{V}.$$

Thus, at any time-step $j \in \mathcal{T}$ and for any module $M_k \in \mathcal{L}$ there must be at most one operation that is executing:

$$\sum_{i} \sum_{h=j-C_i^{M_k}+1}^{j} \sum_{l} z_{i,h,k,l} \leq 1., \quad \forall M_k \in \mathcal{L}, j \in \mathcal{T}. \quad (6)$$

4.1.3 Placement constraints

The allocated modules have to be placed on-chip such that they do not *physically* overlap. However, since a biochip is reconfigurable, the same cell area can be used by two different modules as long as they do not overlap *in time*. Hence, the placement constraints will be expressed as a function of time, considering each time-step j in the schedule.

The first constraint to be considered is the size of the microfluidic array of the biochip. At each time step j, the sum of the modules that are placed on the array should not exceed the total area size, $m \times n$:

$$\sum_{i} \sum_{h=j-C_i^{M_k}+1}^{j} \sum_{k} \sum_{l} z_{i,h,k,l} \times L_k \times W_k \leq m \times n, \forall j \in \mathcal{T} \quad (7)$$

where L_k and W_k are the length and width of module M_k, respectively, measured in number of cells.

The second constraint captures that no modules should overlap, i.e., a cell c_l on the array can be occupied by at most one module during time step t_j.

Let as consider a cell c_r (with coordinates x_r and y_r) which is the top-left corner of module M_k. If cell c_l is within the rectangle formed by M_k, i.e., $x_l - L_k + 1 \leq x_r \leq x_l$ and $y_l - W_k + 1 \leq y_r \leq y_l$, then we have to impose the restriction that no other module is active during this time interval:

$$\sum_{i} \sum_{h=j-C_i^{M_k}+1}^{k} \sum_{k} \sum_{r} z_{i,h,k,r} \leq 1. \quad (8)$$

4.1.4 Fault tolerance constraints

In step two of our general synthesis strategy outlined at the beginning of this section, we are interested to synthesise a reconfigured implementation such that the set $\mathcal{F}$ of faulty cells is excluded during placement. The reconfiguration of an implementation Ψ_i is performed using the same ILP-based approach presented so far. The only difference is that we are constrained by using the same allocation as Ψ_i, and by the fault-tolerance constraint:

$$\sum_{i} \sum_{j} \sum_{k} \sum_{r} z_{i,j,k,r} = 0, \quad (9)$$

where $x_l - L_k + 1 \leq x_r \leq x_l$ and $y_l - W_k + 1 \leq y_r \leq y_l, \forall c_l \in \mathcal{F}$.

5 EXPERIMENTAL EVALUATION

We were interested to evaluate the ILP-based approach proposed in the previous section. For this purpose, we have used a real-life example consisting of the mixing stage of a polymerase chain reaction application (PCR/M), which is one of the most common techniques for DNA analysis. Researchers have shown how PCR can be implemented using digital microfluidic biochips, such as the ones considered in this paper (Chakrabarty and Su 2006).

We have solved the ILP model with GAMS 21.5 using the CPLEX 9.130 solver, running on Sun Fire v440 computers with 4 UltraSPARC IIIi CPUs at 1,062 MHz and 8 GB of RAM. We have considered four area constraints, 5x5, 6x6, 7x7 and 8x8 and the module library in Figure 1(c). The time-limit imposed on the application completion time was δ_G = 13 s. We have implemented the PCR/M application on these architectures using the ILP/S approach proposed in the previous section. The optimum schedule lengths obtained for each area constraint are presented in Table 2, second column. We can see that the timing constraint δ_G is not met for the small area size of 5x5 cells. This is because the small chip size does not allow the placement of enough mixers to explore the parallelism in PCR/M.

Table 2: Experimental results for PCR/M

Area	$\delta_{Optimum}$	% reconfig.	Avg. exec. time
5x5	15 s	–	–
6x6	13 s	48.40	19 min 50 s
7x7	13 s	86.96	36 min 50 s
8x8	13 s	96.25	61 min 12 s

Out of these four implementations, we were interested, in the next experiments, to determine which one has the highest probability to be reconfigured successfully in case of faults. Thus, we have generated fault scenarios $\mathcal{F}$ using Monte Carlo Simulation, considering 5,000 runs and a cell reliability of R_{cell} = 0.999. The reconfiguration in case of a fault scenario consisting of the set $\mathcal{F}$ of faults has been performed using the ILP/R, presented in the previous section. ILP/R considers the following constraints: (i) the same allocation has to be used as determined by ILP/S for the chip, (ii) the set of faulty cells $\mathcal{F}$ cannot be used during placement and (iii) the imposed time-limit on the application time is δ_G = 13 s.

The percentage of successful reconfigurations is presented in column three of Table 2, while the last column presents the average execution time of ILP/R. We have not performed reconfigurations for the 5x5 area, since we were not able to meet the timing constraint in case of no-faults. We can see that as the area constraint is relaxed, we are able to increase the reconfiguration probability from 48.40% to 96.25%. Note that for the 8x8 area, we were not able to obtain

100% reconfigurability because in 3.75% of cases the solver has reached its iteration limit and no implementation was produced.

Using this proposed ILP framework, the designer will be able to explore several design alternatives, and to chose that particular implementation which has the desired area, schedule length and successful reconfiguration probability. For PCR/M, the implementation with the area of 7x7 and an application completion time of 13 s looks most promising, since the area is smaller than 8x8, with the same application completion time and with a comparable reconfigurability probability of 86.96%.

6 CONCLUSION

In this paper we have addressed the synthesis of microfluidic-based biochips, which are based on the manipulation of liquids not as a continuous flow, but as discrete droplets, and hence are highly reconfigurable and scalable.

We have modeled a biochemical application using a acyclic polar graph, where each node is an operation and the edges represent dependencies between the operations. We have proposed an ILP-based synthesis methodology for the allocation, placement, binding and scheduling of operations on the biochip. Using a polymerase chain reaction application we have shown that our ILP-based approach can successfully synthesize the application and find the optimal schedule length under given area constraints.

We have considered multiple faults in the biochip, and we have used Monte Carlo simulation to determine the probability of successful reconfigurability of a certain implementation, such that the faulty cells are not used and the area and timing constraints are satisfied. As the experimental section has shown, our ILP-approach methodology is able to successfully reconfigure the chip with a high probability.

REFERENCES

Avizienis, A., J.-C. Laprie, and B. Randell (2001). Fundamental concepts of dependability. Technical Report 1145, LAAS-CNRS.

Chakrabarty, K. and F. Su (2006). *Digital Microfluidic Biochips: Synthesis, Testing, and Reconfiguration Techniques*. Boca Raton, FL: CRC Press.

Chakrabarty, K. and J. Zeng (2005). Design automation for microfluidics-based biochips. *ACM J. on Emerging Technologies in Comput. Syst. 1*(3), 186–223.

Cho, M. and D. Z. Pan (2008). A high-performance droplet router for digital microfluidic biochips. In *Proc. Int. Symp. Phys. Des. (in press)*.

Deb, N. and R. D. Blanton (2000). Analysis of failure sources in surface-micromachined mems. In *Proc. Int. Test Conf.*, pp. 739–749.

Fair, R. B. (2007). Digital microfluidics: is a true lab-on-a-chip possible? *Microfluidics and Nanofluidics 3*(3), 245–281.

Gajski, D. D., N. Dutt, and S. L. A. Wu (1992). *High-Level Synthesis: Introduction to Chip and System Des.* Kluwer Academic Publishers.

Kerkhoff, H. G. (2007). Testing microelectronic biofluidic systems. *IEEE Des. Test Comput. 24*(1), 72–82.

Micheli, G. D. (1994). *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Science.

Pollack, M. G., A. D. Shenderov, and R. B. Fair (2002). Electrowetting-based actuation of droplets for integrated microfluidics. *Lab Chip J. 2*, 96–101.

Ricketts, A., K. Irick, N. Vijaykrishnan, and M. Irwin (2006). Priority scheduling in digital microfluidics-based biochips. In *Proc. Des., Automat. and Test in Europe Conf.*, Volume 1, pp. 1–6.

Su, F. and K. Chakrabarty (2004). Architectural-level synthesis of digital microfluidics-based biochips. In *Proc. Int. Conf. Comput. Aided Des.*, pp. 223–228.

Su, F. and K. Chakrabarty (2006). Module placement for fault-tolerant microfluidics-based biochips. *ACM Trans. Des. Automat. Electron. Syst. 11*(3), 682–710.

Su, F., W. Hwang, and K. Chakrabarty (2006). Droplet routing in the synthesis of digital microfluidic biochips. In *Proc. Des., Automat. and Test in Europe Conf.*, Volume 1, pp. 73–78.

Thorsen, T., S. Maerkl, and S. Quake (2002). Microfluidic largescale integration. *Sci. 298*, 580–584.

Ullman, D. (1975). Np-complete scheduling problems. *J. Comput. Syst. Sci. 10*, 384–393.

Verpoorte, E. and N. F. D. Rooij (2003). Microfluidics meets mems. *Proc. IEEE 91*, 930–953.

Xu, T. and K. Chakrabarty (2007). Functional testing of digital microfluidic biochips. In *Proc. Int. Test Conf.*, pp. 1–10.

Zhang, T., K. Chakrabarty, and R. B. Fair (2002). *Microelectrofluidic Systems: Modeling and Simulation*. Boca Raton, FL: CRC Press.