DTIP 25-27 April 2012, Cannes, France

Online Synthesis for Error Recovery in Digital Microfluidic Biochips with Operation Variability

Mirela Alistar, Paul Pop, Jan Madsen
Technical Univ. of Denmark, DK-2800 Kgs. Lyngby
Phone: +45 4525 3475, Fax: +45 4593 0074, email: mali@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. Researchers have presented approaches for the synthesis of digital microfluidic biochips, which, starting from a biochemical application and a given biochip architecture, determine the allocation, resource binding, scheduling, placement and routing of the operations in the application. The droplet volumes can vary erroneously due to parametric faults, thus impacting negatively the correctness of the application. Researchers have proposed approaches that synthesize offline predetermined recovery subroutines, which are activated online when errors occur. In this paper, we propose an online synthesis strategy, which determines the appropriate recovery actions at the moment when faults are detected. We have also proposed a biochemical application model which can capture both time-redundant and space-redundant recovery operations. Experiments performed on three real-life case studies show that, by taking into account the biochip configuration when errors occur, our online synthesis is able to reduce the application times.**

I. Introduction

Microfluidic biochips have the potential to replace the conventional laboratory equipment as they integrate all the functions needed to complete a bioassay. Applications on biochips are considered in areas such as drug discovery, clinical diagnosis, DNA sequencing, protein analysis and immunoassays [1] [2] [3]. The digital microfluidic biochips (DMBs) use discrete amounts of fluids of nanoliter volume, named droplets, to perform operations such as: dispensing, transport, mixing, split, dilution and detection. A DMB is modelled as a two-dimensional array of identical electrodes, see Fig. 1a, where each electrode can hold a droplet. Operations such as mix and dilution are reconfigurable, i.e. they take place on any rectangular area of electrodes, called a module, see the biochip in Fig. 1a.

In order to be executed on a DMB, a biochemical application has to be synthesized. The synthesis consists of five main tasks: *allocation*, during which the needed modules are selected from a module library, *binding* the selected modules to the biochemical operations in the application, *placement*, during which the positions of the modules on the biochip are decided, *scheduling*, when the order of operations is determined and *routing* the droplets to the needed locations on the biochip. Researchers have proposed several approaches to the synthesis of DMBs [3] [4] [5].

While a bioassay is executed on a DMB, the droplets are expected to encounter changes in volume during mixing and split operations. Assuming ideal conditions, when two droplets come together for a mixing operation, the resulting droplet has a volume equal with the sum of the input droplets. After a split operation, the resulting droplets have volumes equal to half of the initial droplet. However, the volume of a droplet can also vary erroneously due to parametric faults, such as an electrode coating fault or unequal actuation voltages. Biochemical applications have high accuracy requirements, such as $\pm 2\%$ for microdialysis applications [6] and $\pm 10\%$ in drug discovery applications [7]. Hence, it is imperative to address the biochemical operation variability which results in erroneous volume variations.

Incipient research has addressed the erroneous volume variation due to parametric faults. Thus, in [8], we focused on the erroneous volume variation after an unbalanced split operation. We have proposed a scheduling algorithm to derive the backup static schedule needed to recover from all combinations of faulty split operations. At runtime, the scheduler will switch to the backup schedules corresponding to the observed error occurrences. The work in [9] addresses the volume variations in all operations. Thus, intermediate droplets of correct volumes are stored at checkpoints. When an error is detected, the stored droplets are used in the recovery subroutine. The locations of the checkpoints and the recovery subroutines are determined offline and stored in a microcontroller memory. If an error is detected at a checkpoint, the microcontroller interrupts the bioassay, and executes separately the recovery subroutine.

In [8] and [9] the error recovery actions are determined *offline*, and are applied online when a fault is detected. In this paper, we propose an *online* synthesis strategy, which, during the execution of the biochemical application, synthesizes a new implementation containing the appropriate error recovery actions whenever errors are detected.

II. System Model

A. Biochip Architecture

In a DMB, a droplet is sandwiched between a top ground electrode and a bottom electrode as shown in Fig. 1b. The droplet is separated from electrodes by insulating layers and it can be surrounded by a filler fluid (such as silicone oil) or by air. Two glass plates, a top and a bottom one, protect

 ISBN:

the DMB from external factors.The droplets are manipulated using the electrowetting-on-dielectric (EWD) principle [10]. For example, in Fig. 1b, if the middle electrode on the bottom plate is turned off, and the left electrode is activated by applying voltage, the droplet will move to the left.

A mixing operation is executed when two droplets are moved to the same location and then transported together according to a specific pattern. A split operation is done by applying concurrently the same voltage on both left and right electrodes, while the middle one remains turned off. Dilution is a mixing operation followed by a split operation. Each operation is executed in a determined biochip area, called a module. The execution time of an operation depends on the dimensions of the module on which the operation is executed. Based on experiments, researchers characterize a module library $\mathcal{L}$, such as on in Table I, which provides the area and corresponding execution time that are needed for each operation.

The biochip also contains non-reconfigurable devices such as input and output ports and detectors. A biochip can have built-in sensors, placed on top of the regular electrodes. For example, a photo-diode detector [11], can be used for detecting the concentration of a droplet, whereas a capacitive sensor [12] can determine the volume of a droplet. The capacitive-sensing circuit used to measure the volume, operates at high frequency (15 KHz [10]), therefore the time needed for sensing can be ignored. On the other hand, the photo-diode detectors needs 5 seconds to measure the absorbance of the product droplet in order to determine its concentration. For this type of detector, a transparent droplet has to be mixed with a reagent to generate a colored analyte. The initial droplet becomes not suitable anymore for subsequent operations.

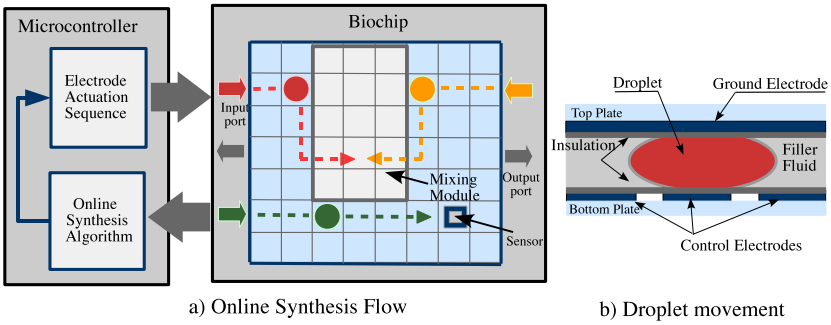
Errors can occur during the execution of fluidic operations due to fabrication defects or malfunctions of the biochip. A complete set of fault models for defects and malfunctions is presented in [13]. Researchers have addressed permanent faults in the context of DMBs [13]. In this paper we focus on parametric faults, which may cause such variations in the volume of the droplets that the bioassay results are erroneous. Each fluidic operation has a specific error range associated with it, named intrinsic error limit, which capture the worst-case volume variations. For example, if the intrinsic error limit E_{Mix} for mixing is 10%, after a mix operation the output droplet can have a volume between 90% and 110% of the nominal value. We use the following notation: E_{Mix} is the intrinsic error limit for mixing operation. E_{Dlt} for dilution, E_{Trans} for transport, E_{Ds} for dispensing, E_{Slt} for split. Experimentally, the following values were determined for the intrinsic error limits: E_{Ds}=E_{Dlt}=E_{Slt}=8%, E_{Mix}=10%, E_{Trans}=12% [9].

B. Biochemical Application Model

A biochemical application is modeled using a acyclic directed graph $\mathcal{G}^0(\mathcal{V},\mathcal{E})$ [4] where the nodes $\mathcal{V}$ represent the operations, and the edges $\mathcal{E}$ represent the dependencies between them. $\mathcal{G}^0(\mathcal{V},\mathcal{E})$ is a polar graph, i.e., it has a *source* node and a *sink* node. Fig. 2a presents an application graph with 7 operations. For example, the node O_5 is a dilution operation, while the node O_7 is a mixing operation. The directed edge between them signifies that operation O_5 has to finish before operation O_7 can start executing. The mixing operation O_7 uses the output droplet issued by dilution operation O_5.

In [9], the authors use error analysis [14] to derive the error limit at the output of an operation from its intrinsic error limit and the limits of the input operations. The equations in Fig. 2d calculate the error limit ε_{Mix} at the output of mixing, ε_{Ds} for dispensing, ε_{Dlt} for dilution, ε_{Trans} for transporting and ε_{Slt} for split operations as a function of intrinsic error limits E_{Mix}, E_{Ds}, E_{Dlt}, E_{Trans} and E_{Slt} respectively, and input error limits I_1 and I_2. The error limit at the output of an operation is propagated and becomes the error limit for its successor operation. In Fig. 2a, for the dilution operation O_5 we have the intrinsic error $E_{Dlt} = 8\%$ and the input operation error limits $I_1 = 8\%$ (for O_1) and $I_2 = 8\%$ (for O_2). Using Eq.(v) from Fig. 2d, we can estimate the error limit at the ouput of operation O_5 to be 17%.

We continue to calculate the error limits for all fluidic operations in the biochemical application. For every bioassay, according to its specific accuracy requirements, the designer decides on a specific volume variation boundary E_{thr}, named threshold error, which is the maximum permitted variation from the nominal volume. When the error, calculated according to the presented error analysis, exceeds the error threshold E_{Thr}, a sensing operation is inserted into $\mathcal{G}^0(\mathcal{V},\mathcal{E})$ to detect if an error occurred or not. For the graph in Fig. 2a, the E_{Thr}



a) Online Synthesis Flow

b) Droplet movement

Fig. 1: Biochip Architecture

TABLE I: Module Library

Operation	Module area	Operation time (s)
Mix	2×5	2
Mix	2×4	3
Mix	1×3	5
Mix	3×3	7
Mix	2×2	10
Split	1×1	0
Storage	1×1	N/A
Sensing	1×1	0

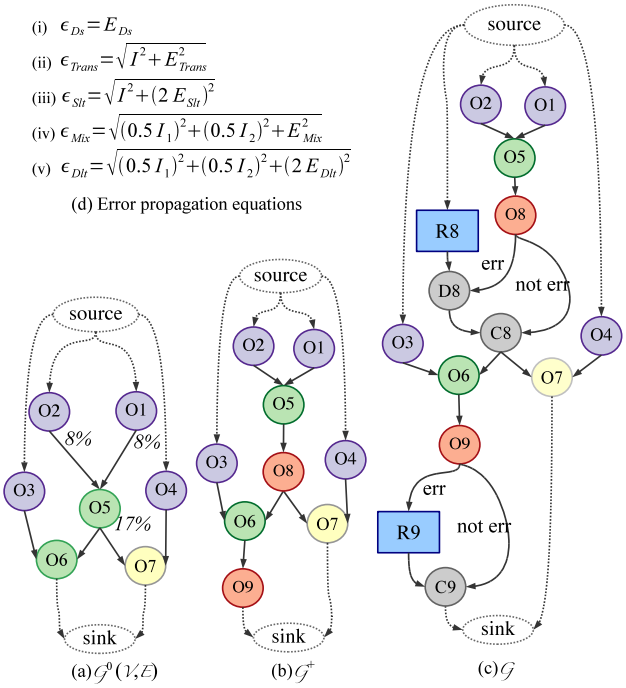


Fig. 2: Example application model

was set to 15%; as a result, the sensing operations O_8, O_9 were inserted into $\mathcal{G}^0(\mathcal{V},\mathcal{E})$, obtaining $\mathcal{G}^+$,as depicted in Fig. 2b.

During a sensing operation the droplet is transported to a capacitive sensor [12] to have its volume measured. Two outcomes are possible after a sensing operation. The first one corresponds to a correct droplet volume, and the second one to an erroneous droplet volume. If the measured volume is outside the expected boundaries, it means that an error occurred. The sensing operation error limit is reset to 0%, since it is assumed that in case an error is detected, the available recovery mechanism is triggered, and the volume of the droplet is brought back to the nominal value.

These two alternative outcomes are represented as conditional edges that connect the sensing operation to the corresponding successor operations. In the graph from Fig. 2c, the left edge connecting sensing operation O_9 and its successor node R_9, is labelled *err* and it corresponds to the case in which the volume sensed was erroneous. The right edge, connecting sensing operation O_9 and its successor node C_9, is labelled *noterr*, which corresponds to the case when the volume sensed was correct. The alternative paths meet in a conjunction node, which, in order to start executing, needs only one of the predecessor operations to finish. In Fig. 2c, node C_9 is at the conjunction of the two alternative paths coming from sensing operation O_9. The conjunction node is a dummy operation which takes no time and needs no space on the biochip.

For each sensing operation, the designer provides a recovery subgraph, that contains all the operations needed to recover from fault. The recovery subgraph is executed whenever a fault occurs and the erroneous droplets are replaced with the droplet resulted from the recovery subgraph. The recovery subgraphs $\mathcal{R}_8$ and $\mathcal{R}_9$ for sensing operations O_8 and O_9 from Fig. 2b are illustrated in Fig. 3b and 3c. In Fig. 2c, the recovery subgraph $\mathcal{R}_8$ is a hierarchical node and it is depicted as a rectangle.

The recovery can be done using time redundancy or space redundancy. The first recovery approach, time redundancy, executes the recovery subgraph after the error has been detected. Space redundancy is a speculative recovery approach, that uses the extra space on the biochip to execute redundantly some of the operations from the recovery subgraph. The tradeoff for using space redundancy is that extra biochip area is needed to execute the recovery subroutine, which can slow down the bioassay completion time, if no errors have occured.

For the time redundancy approach, the recovery subgraph $\mathcal{R}_i$ is inserted into the biochemical application after the sensing operation O_i, on the erroneuos edge. Our approach can handle multiple faults. Thus, after $\mathcal{R}_i$ has finished executing, we re-execute the sensing operation O_i to detect if an error happened during $\mathcal{R}_i$, case is which $\mathcal{R}_i$ is repeated.

The space-redundancy approach is illustrated in Fig. 2c, where the recovery subgraph R_8 from Fig. 3a is connected as depicted in Fig. 2c. As seen in Fig. 2c, if an error is detected during sensing operation O_8, the dummy node D_8 waits for the recovery subgraph $\mathcal{R}_8$ to finish, before it can start executing. In case an error occured during the recovery subgraph, for the space-redundancy approach, then time redundancy recovery is subsequently used. For example, for the space-redundant graph $\mathcal{R}_8$ from Fig. 3a, the time redundancy, denoted with $\mathcal{R}_8(time)$, is inserted after the sensing operation O_8. For O_8 the space-redundant recovery subgraph is depicted in Fig. 3a and the time-redundant one in Fig. 3b. Note that for the same sensing operation, the recovery subgraphs for time and space redundancy may not be identical.

III. Problem Formulation

In this paper we address the following problem. As input we have a biochemical application modeled as a graph $\mathcal{G}^0(\mathcal{V},\mathcal{E})$, which is performed on a biochip platform represented by a

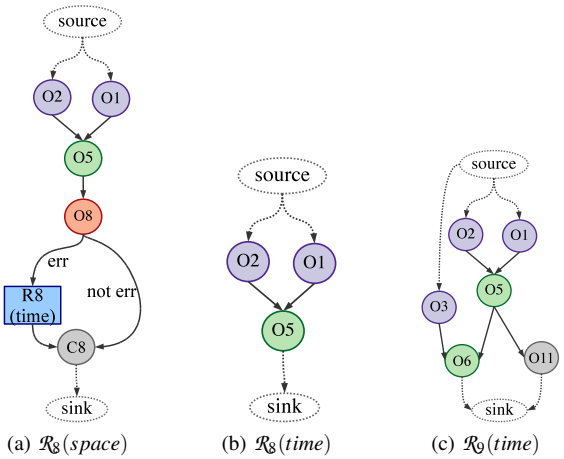


Fig. 3: Recovery subgraphs for O_8 and O_9

$m \times n$ array $\mathcal{C}$ of cells and a characterized module library $\mathcal{L}$. The error threshold E_{thr}, intrinsic operation error limits, and recovery subgraphs $\mathcal{R}_i$ for each sensing operation O_i are also given. We are interested to determine an implementation which minimizes the application completion time in case no errors occur, and at the same time is fault-tolerant to operation variability.

IV. Fault-Tolerant Synthesis Strategy

Our strategy has two components: (1) an offline synthesis algorithm to synthesise the application graph $\mathcal{G}$, such that the application completion time δ_G is minimized and (2) an online synthesis approach which uses a List-Scheduling (LS)-based algorithm, such that the recovery time is minimized. For the offline synthesis, which will also synthesize the space-redundant subgraphs, we use the Tabu Search (TS) based approach from [4]. The offline synthesis tasks performed by TS are presented in the next section. We propose a novel online synthesis (ONS) approach which uses a LS-based algorithm to synthesize online the time-redundant recovery subgraphs.

A. Offline Synthesis

Our strategy takes as input the application graph $\mathcal{G}^0(\mathcal{V}, \mathcal{E})$. We use the error propagation model presented in Section II-B to insert sensing operations into $\mathcal{G}^0$, when the error estimate exceeds the threshold E_{thr}, obtaining $\mathcal{G}^+$. We assume that for each sensing operation the designer has associated a recovery subgraph $\mathcal{R}_i$. These recovery subgraphs are inserted into $\mathcal{G}^+$, as disscused in Section II-B to obtain $\mathcal{G}$. We run the TS-based offline synthesis on the application graph $\mathcal{G}$ and we obtain an implementation Ψ^0 consisting of an allocation $\mathcal{A}^0$, binding $\mathcal{B}^0$, schedule $\mathcal{S}^0$ and placement $\mathcal{P}^0$.

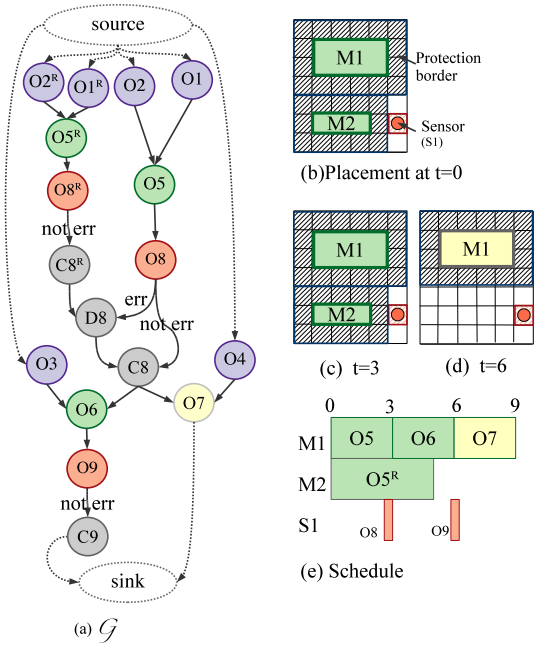


Fig. 4: Offline Synthesis for $\mathcal{G}$ (no errors)

Let us assume that we have to synthesize the graph $\mathcal{G}$ from Fig. 2c on the 7×6 biochip from Fig. 4b. We consider a single capacitive sensor S_1, placed on the biochip as shown in Fig. 4b, where operations O_8 and O_9 will execute. For simplicity, we have ignored the dispensing operations O_1–O_4 in this example. The implementation Ψ^0 is obtained offline under the assumption that no errors have occured. This means that the alternative paths starting with conditional edges labelled *err* are ignored. Thus, for the graph in Fig. 2c we obtain the graph in Fig. 4a, which is then given as input to the TS algorithm from [4] to obtain Ψ^0.

We have to allocate and bind a module for each of the remaining operations: dilution O_5, O_5^R and O_6 and mixing O_7. Considering the module library from Table I, we allocate a 2×6 mixer (M_1) and a 3×5 mixer (M_2) and we bind O_5, O_6 and O_7 to M_1 and O_5^R to M_2. Next, the modules have to be placed on the biochip and the operations scheduled so that the completion time δ_G is minimized. The placement of M_1 and M_2 at time t=0 is depicted in Fig. 4b. Note that when a module in placed on the biochip, a protection border is needed in order to avoid unexpected droplet merging. The schedule is depicted in Fig. 4e, as a Gantt chart, where the operations are presented as rectangles with the length equal to their duration, measured in seconds. As seen in Fig. 4e, operation O_5 and O_5^R start at t=0, O_6 starts at t=3, and sensing operations O_8 and O_9 execute at t=3 and t=6 respectively. The total completion time for the application is 9 seconds. In this paper we ignore the time for droplet transportation.

B. Online Synthesis Strategy

If the volume of a droplet is detected as being outside the imposed threshold, we have to create a new similar droplet with the correct volume (i.e., we recover from the detected error). This can be done in several ways. The Fault-Tolerant Synthesis (FTS) approach proposed in [9] associates to each sensing operation a "recovery subroutine" consisting of a set of operations that have to be executed in order to produce a similar droplet of the correct volume. Thus, the application execution is stopped, the incorrect droplet is discarded, all the other droplets are moved to pre-determined storage areas, and the recovery operations are executed. The allocation, binding, scheduling and placement are for these recovery subroutines are determined offline. For the application from Fig. 2b, considering the recovery subroutines identical to recovery subgraphs $\mathcal{R}_8$ and $\mathcal{R}_9$ from Fig. 3b and 3c, respectively, and the binding and placement determined offline (Fig. 4b), using FTS we obtain a schedule of 11 seconds (Fig. 5a) when a fault is detected by O_8 and 14 seconds (Fig. 6a) when a fault is detected by O_9.

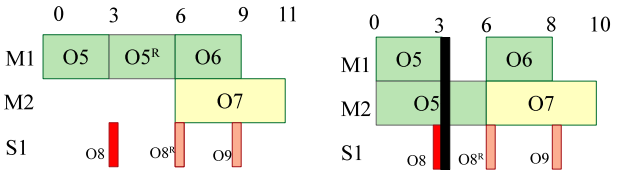
In this paper we propose an online synthesis strategy (ONS) to recover from errors. ONS has the potential to reduce the recovery time compared to an offline approach because it can better exploit the actual biochip configuration, which changes

during the execution as a reaction for potentially multiple errors, at the time when the error occurs. We assume we have a setup as illustrated in Fig. 1a, where the sensors placed on the biochip send the result to the microcontroller (or PC), which controls the biochip. Our ONS approach is invoked after each sensing operation. There are three situations:

(1) The sensing operation detects an error and has a space-redundant subgraph associated to it. This is the situation of O_8 and $\mathcal{R}_8$ in Fig. 2c. The operations in $\mathcal{R}_8$ are executed regardless if an error is detected or not, according to the implementation Ψ^0 derived by TS. If an error is detected, ONS discards the erroneous droplet and waits for the correctly sized droplets produced by the recovery subgraph. For example, in case of error in O_8, we wait in node D_8 for the results from $\mathcal{R}_8$. The obtained schedule is depicted in Fig. 5b; the completion time is 10 seconds, better than using the FTS approach (11 seconds).

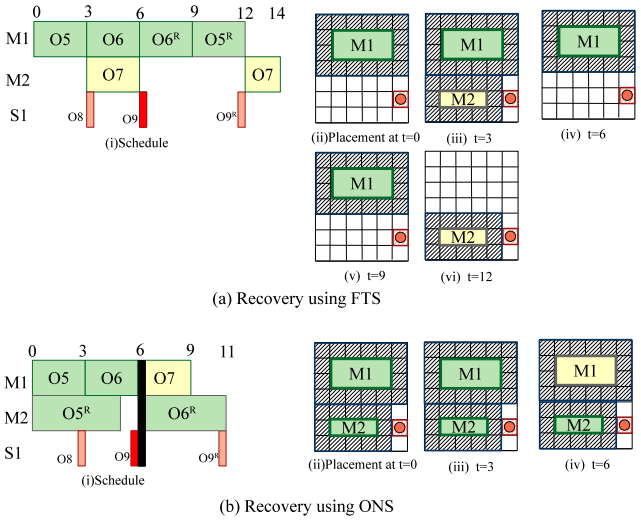
(2) The sensing operation detects an error and has a time-redundancy subgraph associated with it. Such an example is O_9 and the associated subgraph $\mathcal{R}_9$. ONS takes the following steps: first, it stops the execution of the application $\mathcal{G}$; second, it builds a new application graph $\mathcal{G}'$, consisting of the remaining operations from $\mathcal{G}$ (which have not yet started to execute) and the recovery subgraph $\mathcal{R}_i$ associated with sensing operation O_i which has detected the error; third, it calls the LS-based algorithm (presented in Section IV-C) to synthesize a new implementation Ψ'; finally ONS replaces the "electrode actuation sequence" (see Fig. 1a) with the new sequence corresponding to Ψ'. Note that Ψ' contains both the recovery required for O_i, and also a new implementation for the remaining operations in $\mathcal{G}$. In our ONS approach we keep the same allocation $\mathcal{A}^0$ produced by TS for those operations, but we synthesize a new binding $\mathcal{B}'$, schedule $\mathcal{S}'$ and placement $\mathcal{P}'$ using LS. LS has to be very fast since it is executed online, while the biochemical application is stopped, introducing thus an overhead on the application completion time. ONS will try to reudce the recovery time by reusing existing redundant droplets, if available. For example, let us assume that no error was detected by O_8. In this case, we have two correctly-sized droplets, one from O_5 and one from $O_5^R \in \mathcal{R}_8$. In case an error is detected by O_9, ONS will reuse the droplet from O_5^R instead of executing O_1, O_2 and O_5 again, as required by the associated recovery subgraph $\mathcal{R}_9$.

(3) The sensing operation does not detect an error. In this case, no action is taken; the bioassay continues executing without interruptions.



(a) Recovery using FTS (b) Recovery using ONS

Fig. 5: FTS vs. ONS comparison for error on O_8



(a) Recovery using FTS

(b) Recovery using ONS

Fig. 6: FTS vs. ONS comparison for error on O_9

C. List Scheduling-based Synthesis

Every node from $\mathcal{G}$ is assigned a specific priority according to the critical path [15]. All operations ready to run are inserted in a ready list, sorted by priority. The algorithm takes each ready operation O_i from the list, and selects the module with the fastest completion time, that can be accomodated on the boichip. For operation O_6 from $\mathcal{R}_9$, which is ready to be executed at t=4, module M_1 is the fastest option and it is placed on the biochip using the fast-template placement algorithm from [16]. Operation O_6 is scheduled at current time, t=4. We obtain a total completion time of 6 seconds.

V. Experimental Results

In order to evaluate the proposed online synthesis approach, we have used three real life examples. ONS was implemented in Java (JDK 1.6), running on a MacBook Pro computer with Intel Core 2 Duo CPU at 2.53 GHz and 4 GB of RAM. The module library used for all experiments is shown in Table I.

We were interested to determine the gain by doing an online approach to recover from failures, compared to an offline fault-tolerant approach such as FTS [9]. The results obtained are presented in Table. II.

We have used three real-life applications: (1) the mixing stage of polymerase chain reaction (PCR) [17], used for DNA fragments amplification, which has 21 nodes; (2) in-vitro diagnostics on human physiological fluids (IVD) [18], which has 22 operations; (3) the colorimetric protein assay (CPA) [19] utilized for measuring the concentration of a protein in a solution, which has 95 nodes. For all benchmarks we ignored detection operations (used at the end to determine the result of the bioassay) and the input operations. The size of the biochips used to implement these benchmarks is presented in column 2 in Table. II.

For each application we used very low error thresholds (see column 3), which means that the application is very sensitive to volume variations. This has resulted in a graphs with a large number of sensing operations, see column 4, introduced

TABLE II: Experimental Results

App.	Area	E_{Thr} (%)	Sensing Ops.	TS($\mathcal{G}^0$) (s)	FTS (s)	ONS (s)
PCR	7×7	9	7	12	min 14 max 17 avg 14.92	min 12 max 14 avg. 13.85
IVD	7×7	10	9	15	min 17 max 22 avg 19.56	min 15 max 19 avg. 16.44
CPA	10×10	15	39	36	min 38 max 50 avg 40.82	min 38 max 43 avg. 39.34

according to our analysis (we have used the intrinsic error limits introduced in Section II-A). For such sensing operations we have associated a recovery subgraph, as discussed in Section II-B). We have used both time-redundant and space-redundant recovery subgraphs, trying to find a good balance[1] between time and space redundancy. We have built a simulator that can inject faults during the sensing operations. Our approach can handle multiple faults during the execution of the biochemical application. However, to facilitate a comparison to FTS, we have only considered scenarios where a single fault occurs. Columns 6 and 7 in Table II present the results obtained by FTS and ONS respectively, in terms of the resulted end-to-end application completion time. We have run enough simulations to cover all the single-fault scenarios for each benchmark. The times reported for FTS and ONS are the shortest completion time (min), longest completion time (max) and average completion time (avg.) over all the simulations runs.

As we can see from Table. II, we have obtained, for all the applications, better results using our ONS approach than compared to FTS. The results show that using an online approach to fault-tolerance we can obtain better results. Note that we have taken into account the overhead due to the running online ONS, while the application is stopped, which is between 8 to 40 ms, considering that the biochip is connected to a MacBook Pro computer with Intel Core 2 Duo CPU at 2.53 GHz, which runs our ONS strategy. Another measure of the quality of a fault-tolerant algorithm is the overhead introduced due to error recovery. Thus, we have compared the offline results obtained by TS using as input $\mathcal{G}^0$, which does not consider fault-tolerance (no sensing and no recovery), see column 5 in Table II. The average overhead added by ONS, in the fault scenarios we considered, is 15.4% for PCR, 9.6% for IVD and 13.38% for CPA.

VI. Conclusions

In this paper we have presented an online synthesis approach for the synthesis of fault-tolerant biochemical applications. We have addressed digital microfluidic biochips, where the liquids are manipulated using droplets. We have taken into account the parametric faults which can result in operation variability, such as volume variations. We have proposed a biochemical application model which captures the sensing operations needed to detect an error, and the subgraphs that have to be executed for recovery. The sensing operations are introduced based on an error propagation analysis and our model is general enough to capture both time- and space-redundant subgraphs. We have developed a List Scheduling-based fast online synthesis, which is able to exploit the biochip configuration at the moment when errors occur, such that the application completion times, even in case of errors, are minimized. The experiments performed on three real-life case studies show the advantages of the proposed online synthesis heuristic.

[1]The decision between time vs. space-redundancy is an optimization problem which will be tackled in our future work.

References

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

[2] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab Chip*, vol. 4, pp. 310–315, 2004.

[3] K. Chakrabarty, R. B. Fair, and J. Zeng, "Design tools for digital microfluidic biochips: Toward functional diversification and more than Moore," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 7, pp. 1001–1017, July 2010.

[4] E. Maftei, P. Pop, and J. Madsen, "Tabu search-based synthesis of digital microfluidic biochips with dynamically reconfigurable non-rectangular devices," *Design Automation for Embedded Systems*, vol. 14, pp. 287–307, 2010.

[5] T.-W. Huang, S.-Y. Yeh, and T.-Y. Ho, "A network-flow based pin-count aware routing algorithm for broadcast-addressing ewod chips," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1786 – 1799, December 2011.

[6] D. Rose, "Microdispensing technologies in drug discovery," *Drug Discovery Today*, vol. 4, no. 9, pp. 411–419, September 1999.

[7] H. Ren, R. B. Fair, and M. G. Pollack, "Automated on-chip droplet dispensing with volume control by electro-wetting actuation and capacitance metering," *Sensors and Actuators B*, vol. 98, pp. 319–327, 2004.

[8] M. Alistar, E. Maftei, P. Pop, and J. Madsen, "Synthesis of biochemical applications on digital microfluidic biochips with operation variability," *Design Test Integration and Packaging of MEMS/MOEMS (DTIP)*, 2010.

[9] Y. Zhao, T. Xu, and K. Chakrabarty, "Control-path design and error recovery in digital microfluidic lab-on-chip," *ACM Journal on Emerging Technologies in Computing Systems*, 2010.

[10] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip Journal*, vol. 2, pp. 96–101, 2002.

[11] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "A digital microfludic biosensor for multianalyte detection," *Micro Electro Mechanical Systems*, pp. 327 – 330, April 2003.

[12] H. Ren and R. B. Fair, "Micro/nano liter droplet formation and dispensing by capacitance metering and electrowetting actuation," 2002.

[13] T. Xu and K. Chakrabarty, "Fault modeling and functional test methods for digital microfluidic biochips," *IEEE Transactions for Biomedical Circuits and Systems*, vol. 3, pp. 241 – 253, August 2009.

[14] J. R. Taylor, *An introduction to Error Analysis: the study of uncertaincies in physical measurements*. University Science Books, 1982.

[15] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Science, 1994.

[16] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast template placement for reconfigurable computing systems," *IEEE Design and Test of Computers*, vol. 17, no. 1, pp. 68–83, 2000.

[17] M. Kramer and D. Coen, "Enzymatic amplification of dna by pcr: Standard procedures and optimization," *Current Protocols in Molecular Biology*, pp. 15.1.1–15.1.14, 2001.

[18] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proceedings of Design, Automation and Test in Europe*, vol. 1, 2006, pp. 73–78.

[19] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 3, pp. 682–710, 2006.

 ISBN: