Efficient Error Recovery in Cyberphysical Digital-Microfluidic Biochips

Mohamed Ibrahim, *Student Member, IEEE* and Krishnendu Chakrabarty, *Fellow, IEEE*

Abstract—Due to their emergence as an efficient platform for point-of-care clinical diagnostics, digital-microfluidic biochips (DMFBs) have received considerable attention in recent years. In particular, error recoverability is of key interest in medical applications due to the need for system reliability. Errors are likely during droplet manipulation due to defects, chip degradation, and the lack of precision inherent in biochemical experiments. We present an illustrative survey on recently proposed techniques for error recovery. The parameters of the error-recovery design space are shown and evaluated for these schemes. Next, we make use of these evaluations to describe how they can guide error recovery in DMFBs. By exploiting the flexibility provided by field-programmable biochip designs (FPBs), as in the case of their FPGA counterparts in electronic systems, we present a dynamic adaptation technique for error recovery in practical FPBs with a limited number of available control pins. Using two representative real-life bioassays, we show that the proposed approach can provide rapid error recovery for pin-constrained FPBs and it is more effective than recently published methods that target this problem.

Index Terms—Microfluidics, error-recovery, cyberphysical systems, pin-constrained design

1 Introduction

Point-of-care (POC) tests have the potential to improve the management and treatment of infectious diseases, especially in resource-limited settings in developing countries where health care infrastructure is inadequate [1]. In these settings, POC tests can be simply used at the primary-care level with no laboratory infrastructure [2]. In addition, POC tests can potentially empower patients to self-test in the privacy of their homes, especially for diseases such as HIV that carry some degree of social stigma [3]. These issues have motivated researchers from various domains to focus on developing robust technologies for POC diagnostics. Nowadays, clinical pathologists work in cooperation with biochemists to develop new sequences and protocols for obtaining rapid screening results [4]. New substrate materials and biosensors facilitate the miniaturization of sample preparation and automation of POC sequencing on a chip [5]. An example of an emerging technology that has achieved remarkable success in miniaturizing POC testing is digital microfluidics, resulting in digital-microfluidic biochips (DMFBs) [6].

Digital-microfluidic biochip technology, which allows us to manipulate droplets of picoliter volumes under program control on a patterned electrode array, is revolutionizing laboratory procedures not only for point-of-care clinical diagnostics [6], but also for many other applications such as environmental monitoring [7] and drug discovery [8]. Over the past decade, there has been a considerable amount of research on various aspects of automated biochip design and optimization [9], [10], including techniques for architectural-level synthesis [11], [12], module placement [12], and droplet routing [13], [14], [15], [16], [17]. Solutions have also been proposed for co-optimization related to these design problems [18], [19]. For instance, in [18], Liao and Hu have considered co-optimizing module placement and droplet routing while handling variations in biochemical reactions, contamination, and defects resulting from the fabrication process. On the other hand, Keszocze et al. have utilized the power of solvers for *Boolean satisfiability* (SAT) to seamlessly integrate all the design flow steps and optimally generate a one-pass synthesis solution [19]. The majority of these techniques, however, address offline synthesis for application-specific DMFBs. A major stumbling block in the monitoring and controlling of diseases/contaminations via POC systems, for example, is the lack of adaptive and reliable diagnostic tests that can recover from unexpected errors. Moreover, due to the inherent randomness and complexity of the component interactions that are ubiquitous in biochemistry, it is necessary to verify the correctness of on-chip fluidic interactions during bioassay execution [20]. As a result, in order to enhance the market share of POC clinical diagnostic instrumentation, an efficient design of a DMFB requires careful consideration of error recoverability.

A DMFB device is said to have a failure if its operation does not match its specified behavior. In order to detect defects using electrical methods, fault models that efficiently represent the effect of physical defects at some level of abstraction have been described in [21]. These models can be used to capture the effect of physical defects that produce incorrect behavior. Faults can be caused by manufacturing imperfections, or by degradation during use as electrodes are actuated (*i.e., operational fault*). Some possible causes of physical defects are listed below.

- *Dielectric breakdown:* High voltage levels during actuation cause dielectric breakdown, which creates

- *The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708.*
  E-mail: mohamed.s.ibrahim@duke.edu, krish@ee.duke.edu.

Manuscript received 28 Jan. 2015; revised 15 Apr. 2015; accepted 18 Apr. 2015. Date of publication 16 Sept. 2015; date of current version 7 Oct. 2015.
Recommended for acceptance by P.P. Pande.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMSCS.2015.2478457

a short between a droplet and the underlying electrode. In this case, droplet transportation cannot be controlled since the droplet undergoes electrolysis.

- *Degradation of the insulator:* The actuation of electrodes for long durations causes charge to be irreversibly concentrated near the electrodes (*i.e.*, trapping of charge [22]). This causes an operational error since it impedes droplet transportation because of the undesired variation of interfacial surface tension along the droplet flow path.
- *Short-circuited electrodes:* A short between two adjacent electrodes effectively forms one longer electrode. When a droplet resides on this electrode, it is no longer large enough to overlap the gap between adjacent electrodes. As a result, the actuation of the droplet can no longer be achieved.
- *Open in the metal connection between the electrode and the control source:* This open can be on the path between the electrodes and the chip pads, or through the external wires between the biochip and the adjacent controller. This defect results in a failure in activating the electrode for droplet transport.

Besides the physical defects mentioned above, protein fouling may lead to the malfunction of multiple electrodes in the biochip since it renders the surface permanently hydrophilic [23]. This fault, like other operational faults, are hard to predict a *priori* since they often occur during bioassay execution. Our error-recovery system focuses on such types of errors, which must be handled adaptively via a cyberphysical system design.

The correctness of bioassay outcomes can be determined by utilizing on-chip detectors. In addition, physical-aware control software can be used in a DMFB platform to implement an error-recovery method. The work described in Zhao et al. [24] and Luo et al. [20] represent the state-of-the-art for dynamically adapting to error occurrences during assay execution. However, despite their benefits, these methods suffer from several drawbacks:

- The method described in [24] uses checkpointing and rollback recovery, but it is not effective for recovering from errors caused by trapped charge since it continues to use all the electrodes as part of its recovery procedure. The reuse of faulty electrodes can lead to the repeated occurrence of the same errors.
- The approach in [20] carries out re-synthesis of the bioassay to re-execute specific fluidic operations and it can avoid the reuse of faulty electrodes; however, as in [24], it assumes an impractical direct-addressing scheme for electrode addressing, where the number of control pins equals the number of electrodes.
- Both prior methods make an implicit assumption that error recovery is always possible. In practice, however, the number of available electrodes limits the extent to which error recovery is possible.

To overcome these drawbacks, we extend our study of error-recovery schemes by presenting a more flexible error-recovery solution based on FPBs. This approach allows us to achieve error tolerance with a smaller number of control pins, and with array sizes (measured in terms of the number of electrodes) that can be determined *a priori*. The main contributions of this paper are as follows:

- We give an illustrative survey on recently proposed techniques for error recovery. The parameters of the error-recovery design space are shown and evaluated for these schemes.
- We make use of these evaluations to describe how they can guide error recovery in DMFBs.
- We present a dynamic adaptation technique for error recovery in FPBs with a limited number of available control pins. For the same chip footprint, the bioassay completion time in the presence of errors is comparable to that using previous methods that require a much larger number of control pins.
- We present an efficient online synthesis flow to recompute new schedules, module placements, and droplet routes on-the-fly in response to errors.
- We carry out recoverability analysis in order to determine the amount of FPB resources needed for error recovery. This analysis, which we report for a varying number of errors, provides guidelines on how large an array should be used for the target application.

The rest of the paper is organized as follows. Section 2 discusses the key parameters underlying effective error recovery; this discussion leads to the drawing of the error-recovery design space. The available biochip resources and the control system are considered as guidelines to identify appropriate error-recovery components. The process of designing error-recovery components is described in Section 3. A real-life experiment for error recovery based on a biochip that was fabricated in our laboratory, coupled with a sensor circuit, is illustrated in Section 6. In Section 5, we present a reconfiguration technique and its associated online synthesis flow for error-recovery in FPBs. Experimental results for this reconfiguration technique and comparisons with prior work are presented in Section 6. Finally, Section 7 concludes the paper.

2 Design Considerations for an Error-Recovery Technique

To assist with verifiable bioassay execution in POC clinical diagnostic settings, researchers have recently presented a set of error-recovery schemes that are suitable for DMFBs [20], [21], [24], [25]. These schemes vary in their design characteristics with respect to resource utilization and response time. Although these proposals have not provided formal methods for design-space exploration, tradeoffs between error-recovery (response) time, area overhead, control-memory usage, and scalability can be inferred based on their methodologies. Accordingly, multidimensional design-space exploration is required in order to help system designers to pinpoint recoverability limitations, potentially using statistical analysis, for a given configuration.

We begin our design-space exploration by introducing some formal definitions that represent the various

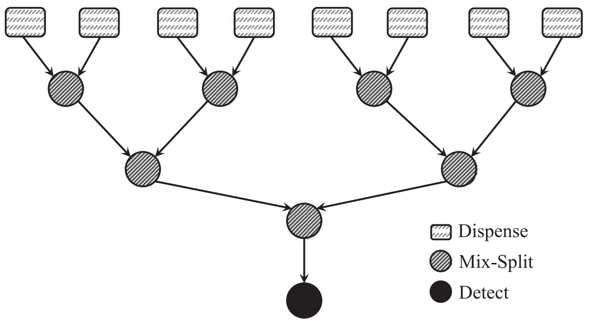

Fig. 1. A sequencing graph on PCR for DNA amplification.

optimization knobs. Then, we evaluate the methods proposed in [20], [21], [24], [25] in an objective manner using these definitions.

2.1 Responsiveness

At the time when an error occurs, the control software requires a period of time, called *recovery time* T_r, to recover from the error and ensure error-free execution. This time is a measure of system responsiveness; *i.e.*, the shorter the time, the better the responsiveness. Precisely, this period of time consists of two phases: 1) *Recovery-procedure triggering time* (T_g), which is the duration between error occurrence and error detection; 2) *Recovery-procedure execution time* (T_e), which is the time required by the procedure to ensure error-free execution from the same (previously erroneous) operation. Note that the system responsiveness R is inversely proportional to the recovery time.; *i.e.*, $T_r = T_g + T_e$ and $R \propto \frac{1}{T_r}$.

Typically, the triggering time T_g depends on the technology of the sensor attached to the system. Deploying a fluorescence detector such as in [24] or a capacitive sensor as in [21] forces the system designer to insert checkpoints to detect errors. During execution, droplets need to be driven into these locations for verification. Hence, it is possible that an error occurs at a certain time but it is not detected until the droplet is driven subsequently to a checkpoint, thereby making T_g relatively long. On the other hand, the use of a CCD camera-based sensing system leads to negligible triggering time since an error can be detected immediately, as shown in [20], [25].

In [20], Luo et al. provided a study to compare the efficiency of CCD camera-based sensors with that of optical sensors. It was shown that a control system that is equipped with a CCD camera-based sensor is able to complete the protocol execution (including error-recovery) with less completion-time (i.e., it is more efficient). Despite the fact that a CCD camera-based sensor provides higher responsiveness, Luo et al. pointed out that optical detectors must be selected for experiments that include photosensitive samples or reagents [26].

The recovery-procedure execution time T_e is linked to the class of recovery system coupled with the chip. In [24], Zhao et al. provide control paths in which the insertion of detection checkpoints and the generation of copy droplets are embedded in the initial synthesis result. Therefore, T_e is relatively short. A shorter recovery execution time is seen in [25], where the recovery sequence is loaded directly from on-chip memory. The dynamic resynthesis procedure in [20], however, requires more time for dynamic adaptation, making T_e longer.

2.2 Area Overhead

A popular strategy adopted in fault-tolerant computing is spatial redundancy of hardware units to generate reliable outputs. This idea can be exploited for error recovery in a DMFB. During bioassay execution, some backup droplets (also called copy droplets) are kept on the chip array to be used as starting points for the recovery sequence. Therefore, the larger the number of copy droplets, the shorter the recovery sequence. However, the presence of copy droplets at the boundary cells of the array imposes additional space constraints on regular bioassay operations. Therefore, a control system that executes an assay containing N operations (i.e., nodes) and produces C copy droplets is said to have backup ratio BR, where $BR = \frac{C}{N}$.

Note that BR never reaches the value 1 since not all operations can produce copy droplets. For instance, the conventional Polymerase Chain Reaction (PCR) process shown in Fig. 1 (an essential part of any clinical diagnosis protocol) contains eight dispense nodes, seven mix-split nodes, and one detection node. The mix-splits are the only operations that can produce copy droplets. As a result, a control system that enables the storage of all possible copy droplets will have $BR = 7/16$. Note that producing more copy droplets does not necessarily mean shorter recovery time. Copy droplets can cause operations to stall due to space limitations. Therefore, error-recovery systems can also choose to store fewer droplets, resulting in smaller BR.

Another aspect of recovery-space cost can be analyzed on the basis of reliability. Errors such as the generation of droplets with abnormal volumes are usually caused by the accumulation of charge on the surface of certain electrodes [22]. If the use of such electrodes is continued, it is likely that they will introduce more errors. Thus, in order to ensure the reliability of DMFB, the electrodes at which an error has been deemed to have occurred should be bypassed in the new synthesis results [20]. Note that the number of discarded cells due to the reliability requirement is tightly coupled with the type of sensor used in the system. For instance, the systems in [21], [24] use fluorescence detectors and capacitive sensors, respectively. In these systems, the faulty electrode cannot be precisely located. Therefore, *region-level* reliability is adopted by discarding a path (region) of cells that represent the path of the erroneous droplet. On the other hand, a CCD camera-based sensing system is used in [20], [25] which is able to accurately locate the faulty electrode; thus adopting *cell-level* reliability. We define the reliability cost to be the average number of discarded cells per recovery run.

2.3 Control-Memory Usage Cost

Quantification of control-memory usage also depends on the class of recovery system used. Luo et al. in [20] employ an *a posteriori* approach in which a dynamic resynthesis technique is invoked to recover from an error. In this case, no prior recovery information needs to be stored; thus memory utilization is minimized. On the other hand, the work in [25] follows an *a priori* approach such that all the possible recovery sequences for the operations are stored in memory before actual execution. This class targets real-time applications that demand rapid response. Nevertheless, a significant amount of memory-storage is required to cover all error possibilities. There is clearly a tradeoff between the response speed and the memory utilization in error-recovery systems.

2.4 Recovery from Multiple Concurrent Errors

Handling situations when multiple errors occur concurrently is not a trivial problem. Recovering from multiple errors requires not only additional support from the sensing system, but also support from the recovery controller to prioritize recovery sequences [20]. For example, the error recovery scheme in [24] is unable to handle these situations, not because of the fluorescence detectors, but because the offline synthesis can only handle a recovery sequence from one error at a time. In case the fluorescence detectors report the occurrence of multiple errors, the recovery from these errors is performed sequentially. This strategy leads to inefficiencies in the recovery system. On the other hand, the cyberphysical approach in [20], either using fluorescence detectors or CCD camera-based sensors, is able to dynamically resynthesize the bioassay sequence upon error detection. In this situation, multiple recovery processes are triggered at the same time and the control software generates a priority queue for each recovery process. After these priority queues are merged, the control software assigns a priority for each element based on topological sort. From a practical point of view, this feature cannot be easily supported in [25] due to the exponential growth of the amount of memory required to store all possible combinations of errors.

Based on the four parameters defined above, we are able to develop a profile of the parameter space for error-recovery. Based on this reverse-engineering process, we can export this knowledge into a system design engine that selects an appropriate error-recovery scheme for a given configuration.

3 Options for Error Recovery

Limitations in technologies related to CPU processing, memory, biosensor technique, and actuation capability impose constraints on error-recovery capabilities. These limitations motivate careful design in order to fully exploit the parameter space described in the previous section. In this section, we give some examples of how decisions can be taken for designing an error-recovery system that fits a certain configuration.

3.1 Simple Memory-Limited Error Recovery for a Large Chip

3.1.1 Problem Statement

We are given a large array that is intended to be used for rapid testing based on a highly multiplexed protocol, such as DNA sequencing. The chip array is coupled with a standard optical sensor such as a fluorescence detector. The objective is to design a simple error-recovery scheme that uses limited memory storage.

3.1.2 Solution

To provide the targeted DMFB with a suitable error-recovery scheme, we map the aforementioned system attributes into the parameter space. Since the system is equipped with a typical fluorescence detector, we have to incorporate a set of monitoring checkpoints at which verification is performed [24]. Undoubtedly, the system responsiveness in erroreous cases will be adversely affected due to the recovery-procedure triggering time T_g.

With this sensor technology in hand, we have two choices for developing a recovery controller that operates based on the incorporated checkpoints. The first choice is to store the recovery sequences for all the checkpoints in memory. Then, a sequence will be immediately loaded if needed (i.e., on error detection). Note that this controller does not interrupt any of the other ongoing bioassay-related fluidic operations. However, this choice requires a large memory, which is not available in this setting. As a result, instead of storing all recovery sequences in advance, the second choice is to implement a *rollback* recovery that re-executes a portion of the protocol sequence. This rollback approach can be developed by empowering the controller with a dynamic resynthesis ability that produces new resource bindings, module placements, and droplet routings whenever an error is detected [20]. However, resynthesis execution is time-consuming because every component in the synthesis toolchain needs to be invoked. An alternative rollback methodology that does not perform resynthesis, but makes use of the available chip area, is to have a sufficiently large number of copy droplets that can be flexibly transported on-demand and used if needed. This principle is adopted in [24].

The proposed concept of rollback recovery redefines a checkpoint to be a monitoring as well as a copy-droplet storage location. Therefore, if an error is detected at a checkpoint, the rollback procedure will re-execute all fluidic operations from the immediate upstream checkpoint along the paths in the protocol sequence. The re-execution procedure will benefit from the copy droplet stored at this upstream checkpoint. Note that a significant portion of the chip area is utilized now for storing copy droplets for the checkpoints. This cost is, however, justifiable given the large chip size.

3.2 Delay-Tolerant Memory-Aware Error Recovery for a Practical Chip

3.2.1 Problem Statement

From a practical point of view, the array size of the given DMFB is small. Therefore, we have a setting where we are not able to keep many copy droplets on the chip. In some cases, a checkpoint will be used for monitoring without the storage of any copy droplets. Thus, the rollback procedure demands more intelligence in deciding which path of operations needs to be re-executed. Moreover, the on-chip memory storage is also limited.

Fortunately, our system in this scenario is provided with a CCD camera-based sensor. Hence, the objective is to leverage the benefit of CCD camera monitoring to architect an error-recovery scheme that can dynamically respond to

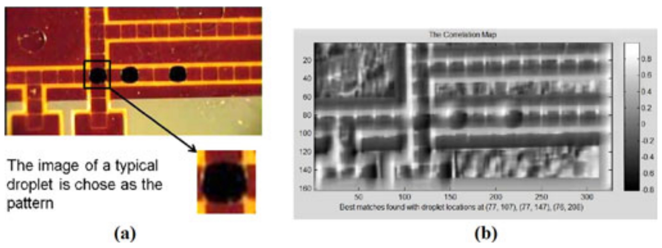
The image of a typical droplet is chose as the pattern
(a)
The Correlation Map
Best matches found with droplet locations at (77, 107), (77, 147), (76, 200)
(b)

Fig. 2. Illustration of CCD camera-based monitoring system. (a) Captured image. (b) Correlation map between the array image and the pattern [20].

errors by considering the memory-storage limitation. The advantage of this setting can be leveraged with complex protocols, such as exponential dilution [24]. In this protocol, dilution operations are performed over several steps, thus an error can occur at any step, i.e., for any targeted sample concentration. Due to the long sequence of dilution operations, it is not feasible to waste a significant amount of chip area to store copy droplets for every step. It is also impractical to pre-store an error-recovery solution for every error scenario for such complex protocols. Since such applications can tolerate a small increase in completion time, a flexible "dynamic" error-recovery approach is the preferred choice for this setting.

3.2.2 Solution

Since the chip is equipped with a CCD camera-based sensor, we no longer need to determine the locations of checkpoints to be inserted during initial design synthesis. The CCD camera can be used in experiments to show the plan view of currently moving droplets [20], as shown in Fig. 2a. Based on the images captured by the CCD camera, droplets can be automatically located by the control software using image processing. The control software generates a correlation map between the array image and a pattern that represents the image of a typical droplet [20], [27]. Fig. 2 shows an image that is captured by the CCD camera as well as the generated correlation map. Obviously, this approach not only enables the control software to detect errors immediately and therefore eliminating the triggering time T_g, but it is also capable of precisely identifying the cell location of the error; i.e., providing cell-level error localization. As a result, the CCD camera can be efficiently used to observe the cyberphysical system. This advantage justifies the additional cost incurred due to the additional sensor instrumentation.

With the availability of on-chip sensors and the hardware that can send feedback to the control software, it is now necessary to design physical-aware software that can analyze sensor data and dynamically adapt to it. Adaptations includes updates for the schedule of fluid-handling operations, resource binding, module placement, and droplet routing pathways. Therefore, based on the work proposed in [20], the task of the recovery controller includes the following two phases: 1) The pre-execution (offline) data preparation phase, which is essentially the initial synthesis of the protocol sequencing graph; 2) The online monitoring and dynamic adaptation phase, which provides an algorithmic resynthesis capability for the controller [20]. Hence, during the execution of the bioassay, detection of an error will trigger error recovery; a backtrace will be performed in the original sequencing graph to identify which portions of the previously executed operations need to be re-executed. In this approach, the backtracing path depends on how many copy droplets are available on the chip.

A drawback of this cyberphysical error-recovery scheme is the delay incurred in adaptation due to the need for resynthesis. Such a resynthesis step also causes other fluid-handling operations to be interrupted. However, compared with the timing of the previous scenario, a small increase in T_e is balanced with the negligible value of T_g. Moreover, the setting here is delay-tolerant. Finally, the on-chip memory usage is relatively larger than for the previous scenario since both the original and updated synthesis results need to be saved. Moreover, data resulting from image-processing steps needs to be handled. Nevertheless, no significant memory storage is needed since the stored resynthesis and image processing data are continually updated during runtime.

3.3 Real-Time (Fast) Error Recovery for a Practical Chip

3.3.1 Problem Statement

Several clinical diagnosis settings demand that the reactions be rapidly carried out and carefully controlled in real-time to produce desired compounds with high selectivity [25], [28]. As in flash chemistry [28], each step requires highly precise time-control of chemical synthesis, which imposes hard timing constraints on error-recovery. In this third scenario, we are equipped with a chip that has large on-chip memory coupled with a CCD camera-based sensor that is able to monitor the status of moving droplets in real-time. The objective is to take advantage of the available memory to develop a fast error-recovery system; i.e., with negligible recovery time T_r.

3.3.2 Solution

Based on the above analysis, we note that it is time-consuming to dynamically resynthesize the chip protocol whenever an error is detected. To support real-time error recovery, we need to pre-compute and store recovery sequences for all errors of interest that can occur during a bioassay. This issue is highlighted in [25]. When an error is detected by on-chip sensors during the execution of a bioassay, the controller can simply look up the recovery solution (known as *dictionary element*) in memory rather than performing online resynthesis. This dictionary-based solution therefore reduces response time. The generation of these dictionary elements can be performed using simulation before the execution of the experiment. Also, to minimize the memory required in this hardware-assisted error-recovery method, a fast compaction/de-compaction technique can be coupled with the finite-state machine (FSM). The compaction and de-compaction procedures do not impact the response time, as shown in [25].

Despite the significantly low response time, the number of generated dictionary elements increases dramatically if multiple-operation error recovery is considered. Therefore, such a system requires substantial on-chip memory storage. In [25], Luo et al. described a dictionary-

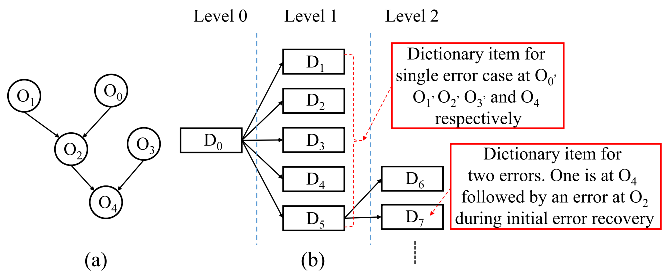
Level 0
Level 1
Level 2
O1
O0
O2
O3
O4
D0
D1
D2
D3
D4
D5
D6
D7
Dictionary item for single error case at O0, O1, O2, O3, and O4 respectively
Dictionary item for two errors. One is at O4 followed by an error at O2 during initial error recovery
(a)
(b)

Fig. 3. Illustration of hardware-assisted error-recovery with multiple error occurrences [20]. (a) An example of a protocol sequencing graph. (b) Tree structure of the error dictionary. Entries at the second level correspond to the possible cases of errors that can occur involving two operation. Tree structure can be extended to provide higher recoverability.

based methodology for pre-computing and storing recovery solutions for multiple error occurrences. As mentioned above, an error-recovery solution from single error occurrences is loaded from a list of dictionary elements. This list, however, needs to be extended to a tree structure in order to cover the possible cases of errors that can occur involving multiple operations, as shown in Fig. 3. For instance, if the execution of the protocol in Fig. 3a is error-free, then dictionary item D_0 will only be used. If an error is detected during the execution of operation O_4, then dictionary item D_5 will be loaded for error recovery. During error recovery, if another error is detected during the execution of operation O_2, then dictionary item D_7 will be immediately loaded. Note that every stored dictionary element corresponds to a bundle of actuation sequences for all chip electrodes.

Besides the memory storage challenge, the availability of copy droplets at specific locations must be communicated to the recovery controller in advance such that the droplet

ASU: Array Space Utilization, **RET:** Recovery Execution Time,
RTT: Recovery Triggering Time
MS: Memory Storage, **CED:** Concurrent Error Detection

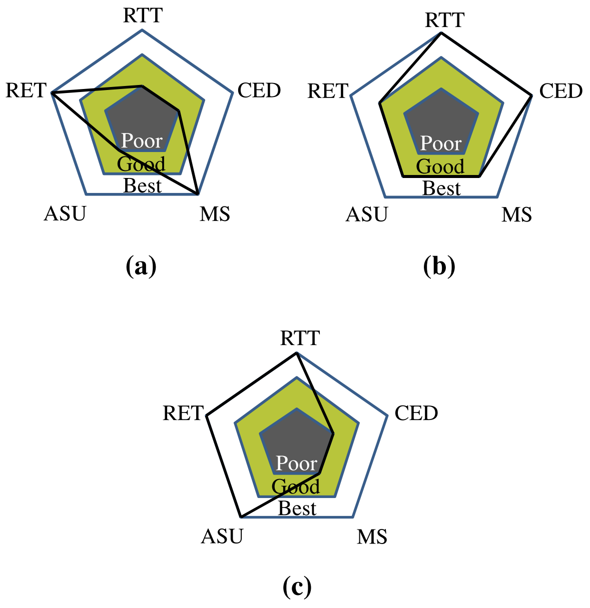

Fig. 4. Illustration of the tradeoffs apparent in the following recovery systems: (a) checkpoint-based offline error recovery [24], (b) software-based cyberphysical error recovery [20], and (c) dictionary-based real-time cyberphysical error recovery [25].

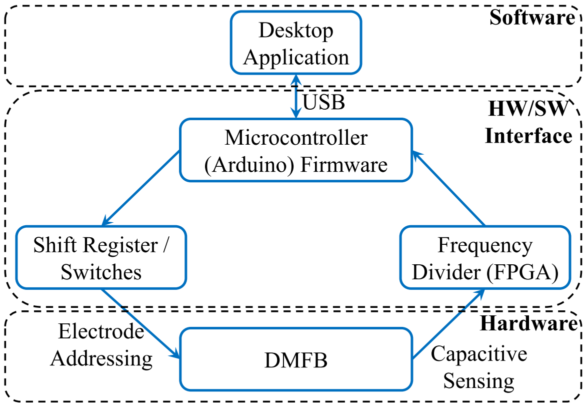

Fig. 5. The setup of the control system loop [21].

can be easily transported and utilized. This information also provides the system with full controllability over the entire chip array.

3.4 Qualitative Comparison

Fig. 4 qualitatively summarizes the tradeoffs inherent in the recovery systems described in the previous sections. The first platform discussed in Section 3.1 is referred to as *checkpoint-based offline error recovery*. The DMFB platform illustrated in Section 3.2 is referred to as *software-based cyberphysical error recovery*. Finally, the fast error-recovery dicussed in Section 3.3 is called *dictionary-based real-time cyberphysical error recovery*.

A key observation is that recovery support for multiple concurrent errors is tightly coupled to the type of on-chip sensor technology. A controller that receives feedback signals from checkpoint-based fluorescence detector is not able to directly support this capability. A CCD camera-based sensor, on the other hand, facilitates execution of such fine-grained monitoring. However, in dictionary-based error recovery, although the CCD camera-based sensor is available, the large number of error combinations that need to be considered *a priori* significantly complicates concurrent multiple-operation error recovery.

4 Laboratory Case Study and Demonstration

As a case study of error-recovery realization, we discuss the recent implementation of a hardware-assisted error recovery platform [21]. We describe an integrated demonstration of cyberphysical coupling in DMFB, whereby errors in droplet transportation on the chip array are detected using capacitive sensors. We also show how the test outcome is interpreted by the controller to autonomously accomplish error recovery as needed.

4.1 Control System Setup

The physical setup for error detection and recovery demonstrates the coordination between the hardware (chip and sensors) and the control software. The hardware/software interface is realized through seamless interaction between control software, an off-the-shelf micro-controller, a shift register for synchronous actuation of 32 chip electrodes, and a frequency divider implemented on an FPGA [21]. The schematic of the control system is shown in Fig. 5. The intermediate micro-controller provides the following functionalities: 1) communication with the desktop application via a USB link, 2) a parallel-to-serial conversion for the 32-bit electrode actuation vector provided by the desktop and injecting the serial data into the shift register, 3) metering the signal frequency received from the FPGA to interpret the capacitive sensing readout. The details of the sensing circuit will be explained in the following section.

On the hardware side, some electrodes are equipped with the sensing circuit; thus designated as *checkpoints*. For effective error recovery, a number of such checkpoints must be incorporated [21]. However, precise localization of errors is not possible with this sensor technology because the only information available to us is that a droplet is stuck within a region between two checkpoints. Therefore, to ensure reliability whenever an error is detected, a region-level bypassing is employed during control software rollback to recover from the error. Based on this scenario, the chip array is divided into several regions with a checkpoint associated with each region. When a "missing droplet" error is detected at a checkpoint, it can be inferred that the defect lies on the path between this checkpoint and the previous checkpoint on the designated droplet route, thus rollback is initiated. A droplet under test retraces its path from the current checkpoint to the previous checkpoint. Note that the rollback is deemed to have been successful when the previous checkpoint once again reports the presence of a droplet (after a known number of clock cycles). If retracing fails, i.e., the previous checkpoint does not report a droplet, we conclude that the droplet is irreversibly stuck [21]. As a result, a new droplet needs to be dispensed and the cells between the two checkpoints have to be permanently discarded for reliability.

4.2 Capacitive Sensor Technology

To enable the detection of nanoliter scale droplets on a DMFB platform, a capacitance sensing circuit was designed in [21]. The sensor relies on a ring oscillator to test droplet presence (instead of testing droplet volume, which might require a considerable amount of manual intervention). This is based on the inverse relationship between droplet volume V and ring oscillator frequency f; *i.e.*, $V \propto \frac{1}{f}$. It also depends on the ability of a high-speed frequency measurement unit and comparator to classify signals corresponding to the presence or the absence of a droplet [21]. The oscillator circuit is used to monitor the capacitance between the control and ground electrodes in the chip actuator giving an output that is a 0-3.3 V frequency-decoded square wave, ranging from 750 kHz to 1.5 MHz, with a 50 percent duty cycle [21].

4.3 Biochip Specification and Layout

A 32-electrode chip was divided into four regions. The positions of checkpoints are determined to ensure that each region has exactly one checkpoint. From the pre-computed activation sequences, the arrival time for a droplet at a checkpoint is known. Hence, based on the sensor readout, the control software decides whether a droplet has arrived as expected. If a droplet reaches a checkpoint as planned, the experiment will be continued and the droplet will enter the next region. If not, the reconfiguration steps will be triggered automatically, a backup route will be generated in real-time and the new electrode activation sequences will be immediately fed into the biochip.

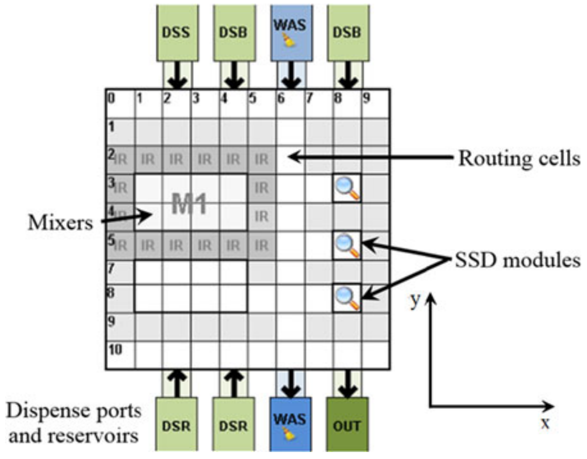

Fig. 6. Overview of an FPB [29].

4.4 Results and Video

Several experimental runs were carried out using the fabricated chip, sensing hardware, the hardware/software interface, and the control software described before. In order to capture a video for the experiment, the researchers in [21] used a CCD camera. Videos illustrate successful rerouting of droplets and bypassing of faults whenever an error is detected.

5 Error Recovery in Pin-Constrained DMFB

In this section, we demonstrate how field-programmable biochips (FPBs) can be used to provide a cyberphysical digital-microfluidic system with efficient error-recovery capability.

5.1 Field-Programmable Biochips

Field-programmable biochips were introduced in [29] to enable the execution of any general sequence of basic bioassay operations, while leveraging the cost-savings of pin-constrained designs. An FPB consists of mixing modules, mixing hold modules, mixing I/O peripherals, Split-Store-Detection (SSD) modules, SSD hold modules, SSD I/O peripherals, and interconnects; see Fig. 6. Note that an FPB requires more control pins than state-of-the-art assay-specific pin-constrained designs [30], [31]. However, in contrast to assay-specific designs, an FPB offers a general-purpose architecture to which any target application can be mapped, and error recovery can be carried out using dynamic resynthesis. Another advantage of this flexible architecture is that it offers operation scalability. It can be extended in the y-dimension (Fig. 6) to increase the number of on-chip modules [29].

From a cyberphysical error-recovery perspective, an FPB has the disadvantage that on-chip components such as Mixing and SSD modules are bound in entirety to bioassay operations; hence, if an erroneous operation involving a component is detected, the entire component must be discarded. This coarse-grained recovery solution is in contrast to directly addressed DMFBs, in which individual faulty electrodes can be bypassed [20]. Nevertheless, the benefits of considerably fewer control pins make FPBs an attractive choice for biochemistry on a chip and dynamic error recovery. Note also that all droplets in an FPB are routed through the electrodes that make up the global routing bus, as shown in Fig. 6. Therefore, if one of these electrodes is faulty, deadlock scenarios can arise in droplet routing. This problem can be addressed by adding redundancy for droplet routing, hence we do not consider it in this work.

5.2 Motivation for Online Synthesis

Online synthesis for error recovery was studied by Alistar et al. in [32]. To cope with the variation in droplet volumes that may cause errors, a redundancy optimization framework was introduced. A study is described in which both time-redundancy (i.e., re-executing erroneous operations) and space redundancy (i.e., creating redundant droplets) are considered for error recovery. If an error is detected during bioassay execution, the proposed system uses the breadth-first search (BFS) technique to traverse the application sequencing graph to generate a recovery subgraph.

Typically, static synthesis for a DMFB includes four steps: scheduling [33], resource binding, module placement [34], and droplet routing[1] [13]. Since an FPB provides a set of field-programmable components to be matched with bioassay operations, resource binding and module placement can be merged into a single step [29].

Without loss of generality, any assay protocol can be executed via the following operations: dispense, mix, split, store, transport and detect. These operations and their dependencies can be depicted in a sequencing graph [12]. Error recovery is coordinated by control software that modifies the sequencing graph to include the fluidic operations needed for recovery. For this purpose, the bioassay operations are formally categorized into three sets:

- **Category I**: this set includes all repeatable bioassay operations (i.e., dispensing and splitting) that can be immediately re-executed when an error occurs.
- **Category II**: this set includes non-repeatable operations (e.g., mixing) that have immediate predecessors as sources of copy droplets. Therefore, they can use these droplets for immediate re-execution when an error occurs. Split/dilution operations are able to provide copy droplets.
- **Category III**: this set includes non-repeatable operations for which the immediate predecessors cannot provide copy droplets. As a result, they cannot be immediately re-executed and they have to stall until a copy droplet is obtained. The control software therefore backtraces to their predecessors in order to deliver the copy droplets.

In order to ensure efficient reconfiguration, the following benefits should be provided by online synthesis:

- *Fast error recovery:* The impact of an error on other operations should be minimized. Rapid recovery leads to the optimization of the bioassay completion time.
- *Reliable error recovery:* The modules at which an error has been deemed to have occurred should be avoided.
- *Copy droplets utilization:* Copy droplets should be stored since they can be used to reduce recovery time.
- *Cost-effective design:* The number of control pins should be minimized.

[1]. Wire routing is considered a chip-level, not physical-level, routing problem.

To satisfy these requirements, we have developed a physical-aware synthesis flow for FPBs. The conventional synthesis flow is adapted for cyberphysical DMFBs by incorporating a CCD camera-based sensing system [20] and a control software interface that initiates resynthesis.

5.3 Control Software Interface

To support dynamic reconfiguration, the software interface must keep track of the "status" of each bioassay operation. The status of an operation includes the following information:

- *ID* and *type*.
- Progress label: The labels are: Not Started, Started, Executed (finished but still delivering droplets to the subsequent operations), Storing (finished but storing copy droplets), or Finished (result droplets are delivered and the operation is retired). It is determined based on the operation's start and finish times.
- On-chip module occupied by the operation.

When an error is detected, the reconfiguration procedure —shown in Algorithm 1—is invoked to immediately generate a new sequencing graph that reflects the error recovery process. It also feeds the subsequent online synthesis algorithm with required data (called "Tokens") for adaptation.

Algorithm 1. Reconfiguration Procedure

```
load original sequencing graph();
$RecoveryPath \Leftarrow$ get ancestors of erroneous op;
$OpsCopyDrops \Leftarrow$ get copy droplets operations;
Optimize recovery ($RecoveryPath, OpsCopyDrops$);
foreach $op$ in operations do
  if $op.progressLabel$ == "NotStarted" then
    $parent \Leftarrow$ get parents of $op$;
    if $parent.progressLabel$ == "Executed" then
      update node title
      ($parent$,"DeliveringDroplet");
    else if $parent.progressLabel$ == "Storing" then
      update node title
      ($parent$,"ExtractCopyDroplet");
  else if $op.progressLabel$ == "Started" then
      update node title ($op$,"ContinuingOp");
  else if $op.progressLabel$ == "Finished" and $op \notin RecoveryPath$ then
    delete node (op);
end
```

To illustrate the generation of the new sequencing graph, we start with the initial schedule of a typical *Protein* assay, as shown in Fig. 7a. Assume, without loss of generality, that an error is detected at the end of **MIX2**. In this case, **MIX1**, **SLT1**, and **MIX7** have finished execution whereas **MIX3** and **MIX8**

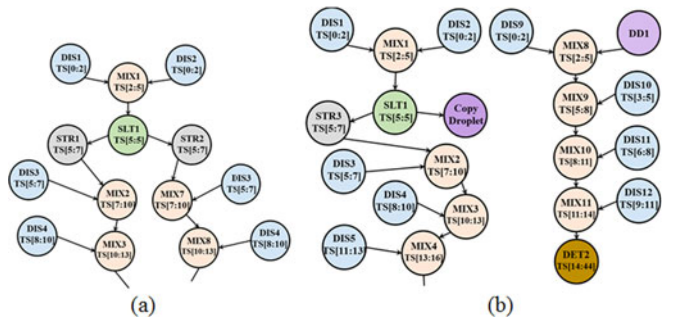

Fig. 7. (a) Initial scheduled sequencing graph for the *Protein* assay. (b) Scheduled sequencing graph when recovery is carried out from an error at MIX2 (the bottom of the sequencing graphs is clipped for lack of space).

have not yet executed. Hence, the reconfiguration procedure would label **MIX1** and **SLT1** as “Finished”. Since **MIX7** is delivering a droplet into the next operation at the moment of error detection, it would be labeled as “Executed”.

Although both **MIX1** and **SLT1** are labeled as “Finished”, they have to be included in the new sequencing graph in order to obtain the required droplet, as shown in Fig. 7b. On the other hand, execution of **MIX7** is not affected by the error at **MIX2**. Since **MIX7** is labeled as “Executed”, it will be replaced by the *Delivering Droplet* node, shown as **DD1** in Fig. 7b. Finally, suppose another error is detected at **MIX4**; the operations **MIX1**, **MIX2**, **MIX3**, **MIX8**, **MIX9**, and **MIX10** are labeled as “Finished” whereas **MIX11** is labeled as “Started”. In this case, however, **MIX1** is not included in the recovery path because a copy droplet is provided by **SLT1**. Thus the node **SLT1** is renamed *ExtractCopyDroplet*. In addition, the node **DET2** in Fig. 7b is replaced with the *ContinuingOp* node to reflect the status of the currently executing operation.

For a bioassay protocol with n fluid-handling operations, the computational complexity of updating the labels of the operations and generating the new sequencing graph is $O(n)$. Therefore, the overhead due to control software adaptation is negligible.

5.4 Adaptive Scheduler

Since error-recovery response time is critical, we use greedy list scheduling [35] (of computational complexity $O(n)$, where n is the number of fluid-handling operations). The list scheduler is invoked when an error is detected. It receives the newly generated sequencing graph and a rescheduling token from the control software interface. This token lists the remaining execution time for each of the *ContinuingOp* nodes such that they can be considered in the new scheduling result. These executing operations are given higher priority.

5.5 Adaptive Placer

In an FPB, there is a topology limit on reconfigurability. A set of cells is dedicated to mix operations whereas other cells are dedicated to store or split operations. This categorization facilitates cost-effective pin-mapping such that mixers, for instance, are controlled with the same set of pins while performing mix operations concurrently [29]. In our work, we use the fast left-edge placement

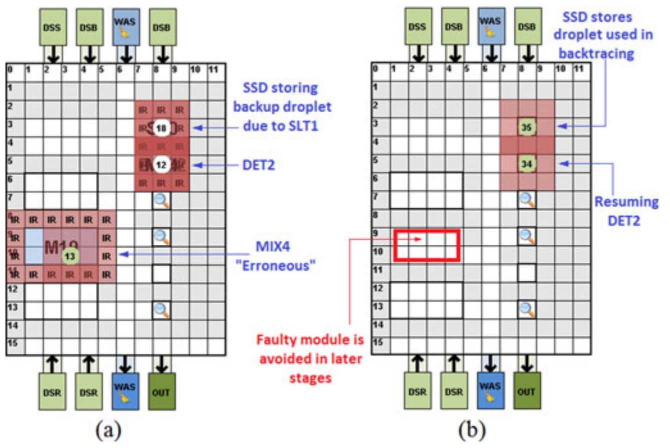
SSD storing backup droplet due to SLT1
DET2
MIX4 "Erroneous"
SSD stores droplet used in backtracing
Resuming DET2
Faulty module is avoided in later stages

Fig. 8. Module placement dynamics: (a) When an error is detected in **MIX4**. (b) On error recovery.

algorithm used in [29]. The computational complexity of this algorithm is $O(n^2)$, where n is the number of fluid-handling operations.

The placer receives a re-placement token when an error is detected. The token describes the modules occupied by the operations represented by the nodes: *ContinuingOp*, *DeliveringDroplet*, and *ExtractCopyDroplet* in order to be considered in the re-placement process. In addition, the copy droplets are placed inside the SSD modules of the array.

As an illustration of placement dynamics, Fig. 8a shows the placement at the moment an error is detected in **MIX4**, as highlighted in Fig. 7a. Two SSD modules are used for storing a copy droplet resulting from **SLT1** and for performing **DET2**, respectively. On error recovery, the token updates the placement for both *ContinuingOp* and *ECD1* leading to the adjusted placement shown in Fig. 8b. In order to ensure reliability, the received token also notifies the placer about the locations of the faulty modules. As a result, modules with errors are not considered in subsequent invocations of the placement algorithm, as shown in Fig. 8b.

5.6 Adaptive Router

It has been shown experimentally that droplet transportation is much faster than fluidic operations such as mixing and splitting [35]. Operation times are in the order of seconds whereas droplet transportation times are in the order of milliseconds. In addition, in an FPB, routing is carried out using a small number of pins to concurrently drive multiple droplets. As a result, instead of using a computationally expensive routing technique for online synthesis, we follow the same approach used in [29], where droplet routing is sequential.

The router uses its knowledge of the location of the erroneous droplet to transport it into the waste reservoir. Next, the router receives a token from the control software interface, which is used to transport the previously stored droplets to their subsequent modules. Based on Fig. 8, the router transports the erroneous droplet resulting from **MIX4** operation into *WAS* port. Then, the router drives a wash droplet among the visited cells by the erroneous droplet to avoid contamination. Finally, the router transports the copy droplet from the SSD module to its destination.

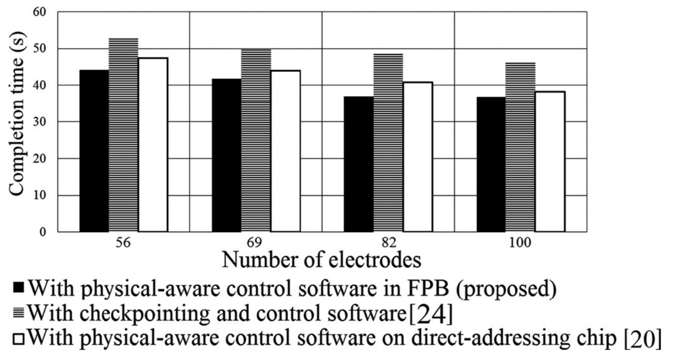

Fig. 9. Assay 1: Completion time in the absence of an error.

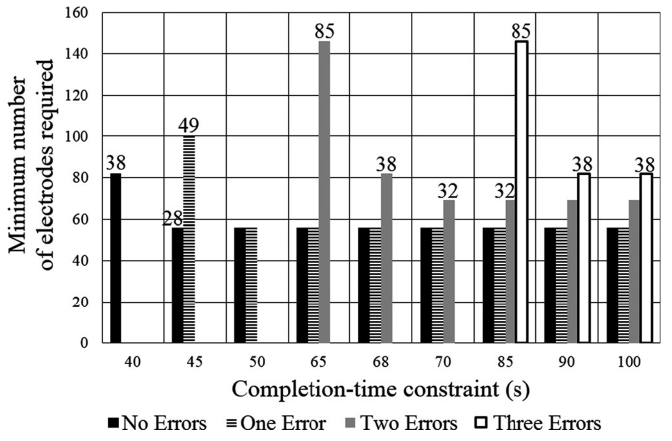

Fig. 11. The minimum number of electrodes required to complete Assay 1 in the presence of completion-time constraint (missing bars indicate that recovery is not feasible; numbers on the bars indicate pin counts).

6 Simulation Results for Error Recovery on Field-Programmable Biochips

We implemented the proposed error-recovery approach using the FPB C++ model described in [29]. The algorithms for the scheduler, the placer, and the router are publicly available in the static synthesis framework in [36]. All evaluations were carried out using a 2.4 GHz Intel Core i5 CPU with 4 GB RAM.

We evaluate the proposed online synthesis framework for FPBs on two representative bioassays that are subject to fluidic errors. We compare our work with two schemes for error recovery: 1) Checkpointing and control paths [24]; 2) Physical-aware control software on a direct-addressing chip [20]. The metrics of comparison include: 1) Error recovery time (in seconds) for various-sized chips; 2) The number of control pins used in each case; 3) The chip size (in number of electrodes) required for recovery. A clock frequency of 100 Hz was considered for the DMFBs [36].

Note that the pre-computed synthesis solution proposed in [25] imposes hard limits on the number of errors for the reasons explained in Section 3.3. As a result, we compare our work with software-based error-recovery methods [20], [24] since these methods are more flexible with respect to the number of errors, and they provide a higher degree of recoverability.

6.1 Preparation of Plasmid DNA (Assay 1)

First, we simulate the preparation of plasmid DNA by alkalinelysis with SDS-minipreparation (Assay 1). The protocols and corresponding sequencing graphs for the preparation of plasmid DNA are described in [37].

To compare the completion times of our method with previous work, we first analyze the error-free execution case; see Fig. 9. As expected, checkpointing leads to completion-time overhead in the absence of errors due to the need for additional fluidic operations. We also note that the cyberphysical FPB that we consider in this work leads to lower completion time than the cyberphysical direct-addressing chip [20], and this difference is especially notable for smaller chip sizes in many cases. This not-so-intuitive result can be attributed to the fact that in the direct-addressing chip considered in [20], the heuristic approach is not effective for resource utilization and for smaller chips, there is considerable serialization due to the smaller size of reconfigurable modules such as mixers. There is also more wait time (stall cycles) for droplet routing due to the sharing of electrodes for routing and mixing.

We next analyze the results when errors are injected randomly into fluidic operations during the execution of the bioassay. The results are shown in Fig. 10. The number of control pins required for various methods is also shown in Fig. 10a. In the presence of errors, the proposed cyberphysical FPB requires completion times that are comparable to that for previous work. More importantly, error recovery can now be achieved using a much smaller number of control pins. For example, single-error recovery can be achieved using an FPB chip with 82 electrodes and 36 pins, while with a direct-addressing chip using the same number of electrodes, 82 pins are used. Note that, due to the substantial reduction in pin-count, error recovery is not

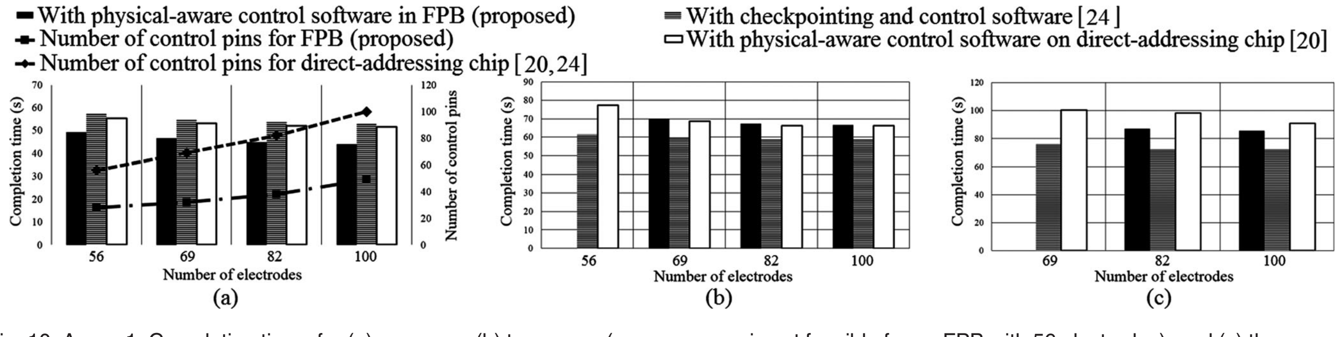

Fig. 10. Assay 1: Completion times for (a) one error, (b) two errors (error recovery is not feasible for an FPB with 56 electrodes), and (c) three errors (error recovery is not feasible for an FPB with 69 electrodes).

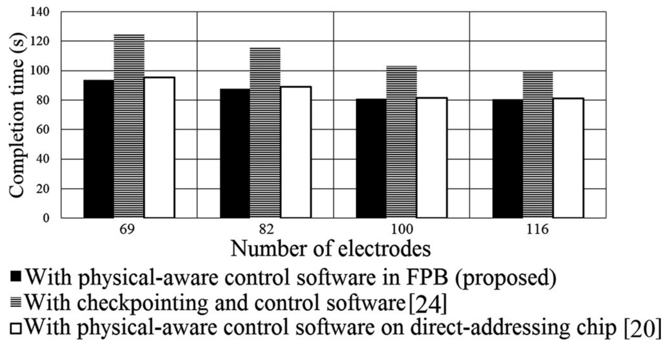

Fig. 12. Assay 2: Completion time in the absence of an error.

possible for an FPB with less than 56 electrodes when there are multiple errors.

We next perform a study on the minimum number of electrodes needed in the proposed FPB design such that Assay 1 completes before a certain time (referred to as completion-time constraint). The results are shown in Fig. 11. This study assumes a worst-case scenario in which the errors are always injected at the protocol level requiring the largest recovery time; for Assay 1, it is the last level in the sequencing graph. Fig. 11 shows that only 82 electrodes are required for fault-free execution within 40 seconds. However, if an error is detected, then the bioassay cannot complete its execution before 40 seconds irrespective of the number of electrodes. Similar electrode-count lower bounds are seen for two and three errors. We note that, compared to the two previous methods, we can achieve error recovery using a much smaller number of control pins. The numbers above the bars in Fig. 11 indicate the number of pins. For [20] and [24], the number of pins equals the number of electrodes.

6.2 Protein Assay (Assay 2)

Next we evaluate the error recovery approach using a protein assay (Assay 2) [20]. Figs. 12 and 13 show that the proposed method provides a completion time that is comparable to the results obtained using prior work, but with a considerably smaller number of control pins.

We then study the minimum number of electrodes required to satisfy a completion-time constraint. Fig. 14 shows that in the presence of errors, the assay cannot complete before 100 seconds. In addition, the assay can complete execution using a small chip with only 69 electrodes in the absence of errors.

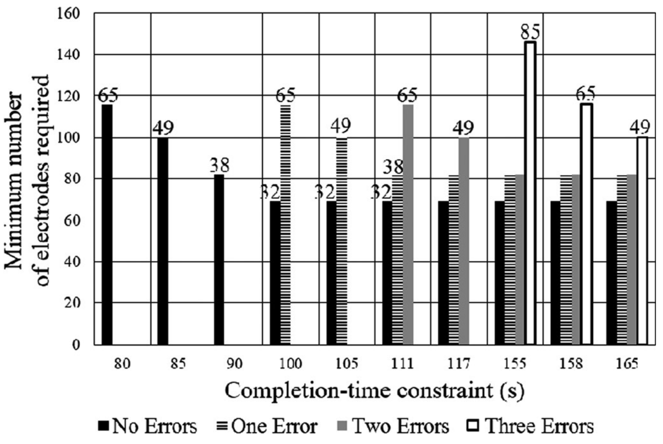

Fig. 14. The minimum number of electrodes required to complete Assay 2 in the presence of completion-time constraint (missing bars indicate that recovery is not feasible; numbers on the bars indicate pin counts).

7 Conclusion

We have described recent proposals for error recovery in a DMBF, which is key for clinical diagnostics applications. We have described four evaluation parameters that form the error-recovery design space. Based on these evaluations, the design of an efficient error-recovery scheme has been outlined for a given a specific system configuration. As a case study, we have described a laboratory setup for a control system that was previously used to provide real-time error recovery for DMFBs. We have also introduced a dynamic adaptation technique based on online re-synthesis for efficient error recovery in field-programmable biochips. This work overcomes a major limitation of prior work, where the constraints on the number of control pins were overlooked in error recovery. The proposed approach has been evaluated using two representative bioassays and compared with two recent techniques for error recovery. We have also reported results on error recoverability analysis for various chip sizes.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant CNS-1135853. A preliminary version of this paper was presented at the IEEE/ACM Design Automation and Test in Europe (DATE), March, 2015. Additional sections were included to reflect our contributions in enhancing the efficiency of error-recovery in cyber-physical digital-microfluidic biochips.

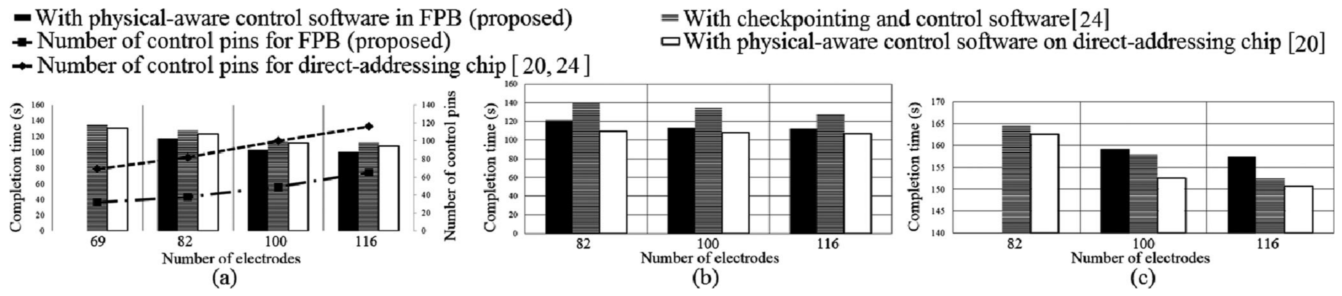

Fig. 13. Assay 2: Completion times for (a) one error (error recovery is not feasible for an FPB with 69 electrodes), (b) two errors, and (c) three errors (error recovery is not feasible for an FPB with 82 electrodes).

REFERENCES

[1] R. Peeling and D. Mabey, "Point-of-care tests for diagnosing infections in the developing world," *Clinical Microbiol. Infection*, vol. 16, no. 8, pp. 1062–1069, 2010.

[2] N. P. Pai, C. Vadnais, C. Denkinger, N. Engel, and M. Pai, "Point-of-care testing for infectious diseases: diversity, complexity, and barriers in low-and middle-income countries," *PLoS Med.*, vol. 9, no. 9, p. e1001306, 2012.

[3] N. Pant Pai and M. B. Klein, "Are we ready for home-based, self-testing for HIV?" *Future HIV Therapy*, vol. 2, no. 6, pp. 515–520, 2008.

[4] K. Lewandrowski, J. Flood, C. Finn, B. Tannous, A. B. Farris, T. I. Benzerx, and E. Lee-Lewandrowski, "Implementation of point-of-care rapid urine testing for drugs of abuse in the emergency department of an academic medical center: Impact on test utilization and ED length of stay," *Am. J. Clin. Pathol.*, vol. 129, no. 5, pp. 796–801, 2008.

[5] C. H. Ahn, J. W. Choi, G. Beaucage, J. H. Nevin, J. B. Lee, A. Puntambekar, and J. Y. Lee, "Disposable smart lab on a chip for point-of-care clinical diagnostics," in *Proc. IEEE*, vol. 92, no. 1, pp. 154–173, Jan. 2004.

[6] R. Sista, Z. Hua, P. Thwar, A. Sudarsan, V. Srinivasan, A. Eckhardt, M. Pollack, and V. Pamula, "Development of a digital microfluidic platform for point of care testing," *Lab Chip*, vol. 8, no. 12, pp. 2091–2104, 2008.

[7] Y. Zhao, S. K. Chung, U.-C. Yi, and S. K. Cho, "Droplet manipulation and microparticle sampling on perforated microfilter membranes," *J. Micromech. Microeng.*, vol. 18, no. 2, p. 025030, 2008.

[8] R. B. Fair, "Digital microfluidics: is a true lab-on-a-chip possible?" *Microfluidics Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.

[9] K. Chakrabarty, "Design automation and test solutions for digital microfluidic biochips," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 4–17, Jan. 2010.

[10] T.-W. Huang, J.-W. Chang, and T.-Y. Ho, "Integrated fluidic-chip co-design methodology for digital microfluidic biochips," in *Proc. Int. Symp. Phys. Des.* 2012, pp. 49–56.

[11] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.* 2004, pp. 223–228.

[12] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2005, pp. 825–830.

[13] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. IEEE/ACM Des. Autom., Test Eur.*, 2006, vol. 1, pp. 1–6.

[14] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow-based routing algorithm for the synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 11, pp. 1928–1941, Nov. 2008.

[15] T.-W. Huang and T.-Y. Ho, "A two-stage integer linear programming-based droplet routing algorithm for pin-constrained digital microfluidic biochips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 2, pp. 215–228, Feb 2011.

[16] Z. Xiao and E. Young, "CrossRouter: A droplet router for cross-referencing digital microfluidic biochips," in *Proc. IEEE Asia South Pacif. Des. Autom. Conf.*, 2010, pp. 269–274.

[17] O. Keszocze, R. Wille, and R. Drechsler, "Exact routing for digital microfluidic biochips with temporary blockages," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2014, pp. 405–410.

[18] C. Liao and S. Hu, "Physical-level synthesis for digital lab-on-a-chip considering variation, contamination, and defect," *IEEE Trans. NanoBiosci.*, vol. 13, no. 1, pp. 3–11, Mar. 2014.

[19] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, "Exact one-pass synthesis of digital microfluidic biochips," in *Proc. IEEE/ACM Des. Autom. Conf.*, 2014, pp. 1–6.

[20] Y. Luo, K. Chakrabarty, and T.-Y. Ho, "Error recovery in cyberphysical digital microfluidic biochips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 1, pp. 59–72, Jan. 2013.

[21] K. Hu, B.-N. Hsu, A. Madison, K. Chakrabarty, and R. Fair, "Fault detection, real-time error recovery, and experimental demonstration for digital microfluidic biochips," in *Proc. IEEE/ACM Des. Autom. Test Eur.*, 2013, pp. 559–564.

[22] H. Verheijen and M. Prins, "Reversible electrowetting and trapping of charge: model and experiments," *Langmuir*, vol. 15, no. 20, pp. 6616–6620, 1999.

[23] V. Srinivasan, V. K. Pamula, P. Paik, and R. B. Fair, "Protein stamping for maldi mass spectrometry using an electrowetting-based microfluidic platform," in *Proc. Optics East*, 2004, pp. 26–32.

[24] Y. Zhao, T. Xu, and K. Chakrabarty, "Integrated control-path design and error recovery in the synthesis of digital microfluidic lab-on-chip," *J. Emerg. Technol. Comput. Syst.*, vol. 6, pp. 11:1–11:28, 2010.

[25] Y. Luo, K. Chakrabarty, and T.-Y. Ho, "Real-time error recovery in cyberphysical digital-microfluidic biochips using a compact dictionary," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 12, pp. 1839–1852, Dec. 2013.

[26] U. Resch-Genger, M. Grabolle, S. Cavaliere-Jaricot, R. Nitschke, and T. Nann, "Quantum dots versus organic dyes as fluorescent labels," *Nature Methods*, vol. 5, no. 9, pp. 763–775, 2008.

[27] Y.-J. Shin and J.-B. Lee, "Machine vision for digital microfluidics," *Rev. Sci. Instrum.*, vol. 81, no. 1, p. 014302, 2010.

[28] J.-I. Yoshida, "Flash chemistry: Flow microreactor synthesis based on high-resolution reaction time control," *Chemical Rec.*, vol. 10, no. 5, pp. 332–341, 2010.

[29] D. Grissom and P. Brisk, "A field-programmable pin-constrained digital microfluidic biochip," in *Proc. 50th IEEE/EDAC/ACM Des. Autom. Conf.*, 2013, pp. 1–9.

[30] C.-Y. Lin and Y.-W. Chang, "ILP-based pin-count aware design methodology for microfluidic biochips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 9, pp. 1315–1327, Sep. 2010.

[31] Y. Zhao, T. Xu, and K. Chakrabarty, "Broadcast electrode-addressing and scheduling methods for pin-constrained digital microfluidic biochips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 7, pp. 986–999, Jul. 2011.

[32] M. Alistar, P. Pop, and J. Madsen. (2015). Redundancy optimization for error recovery in digital microfluidic biochips. *Des. Autom. Embedded Syst.* [Online]. pp. 1–31, Available: http://link.springer.com/article/10.1007/s10617-014-9157-2

[33] D. Grissom and P. Brisk, "Path scheduling on digital microfluidic biochips," in *Proc. IEEE/ACM 49th Annu. Des. Autom. Conf.*, 2012, pp. 26–35.

[34] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," in *Proc. IEEE/ACM 41th Annu. Des. Autom. Conf.*, 2004, pp. 682–710.

[35] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 4, pp. 1:1–1:32, Jan 2008.

[36] D. Grissom, K. O'Neal, B. Preciado, H. Patel, R. Doherty, N. Liao, and P. Brisk, "A digital microfluidic biochip synthesis framework," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr.*, 2012, pp. 177–182.

[37] Y. Luo, K. Chakrabarty, and T.-Y. Ho, "A cyberphysical synthesis approach for error recovery in digital microfluidic biochips," in *Proc. IEEE/ACM Des., Autom., Test Eur.*, 2012, pp. 1239–1244.

Mohamed Ibrahim (S'13) received the BSc (Hons.) degree in electrical engineering in 2010 and the MSc degree in 2013 both from Ain Shams University, Cairo, Egypt. He is currently working toward the PhD degree with the Department of Electrical and Computer Engineering, Duke University, Durham, NC. He was appointed as a research and teaching assistant by the Faculty of Engineering, Ain Shams University, since he received the graduation. In 2014, he joined Prof. Chakrabarty's lab to work on the design automation and test of next-generation cyberphysical digital-microfluidic biochips. His research interests also include mixed-signal very-large scale integration design, microelectromechanical-system modeling, and simulation. He is a student member of the IEEE.

Krishnendu Chakrabarty (F'08) received the BTech degree from the Indian Institute of Technology, Kharagpur, in 1990, and the MSE and PhD degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively. He is currently the William H. Younger distinguished professor of engineering in the Department of Electrical and Computer Engineering and a professor of computer science at Duke University. In addition, he serves as the executive director in Graduate Studies in Electrical and Computer Engineering. He received the US National Science Foundation Early Faculty (CAREER) award, the Office of Naval Research Young Investigator award, the Humboldt Research Award from the Alexander von Humboldt Foundation, Germany, the IEEE Transactions on CAD Donald O. Pederson Best Paper award (2015), and 11 Best Paper awards at major IEEE conferences. He received the IEEE Computer Society Technical Achievement Award (2015) and the Distinguished Alumnus Award from the Indian Institute of Technology, Kharagpur (2014). He current research projects include: testing and design-for-testability of integrated circuits; digital microfluidics, biochips, and cyberphysical systems; optimization of digital print and enterprise systems. He has also led major research projects in the past on wireless sensor networks, embedded real-time operating systems, and chip cooling using digital microfluidics. He has authored 17 books on these topics (with one more book in press), published over 550 papers in journals and refereed conference proceedings, and given over 250 invited, keynote, and plenary talks. He has also presented 40 tutorials at major international conferences. He holds five US patents, with several patents pending. He was a 2009 Invitational fellow of the Japan Society for the Promotion of Science (JSPS). He received the 2008 Duke University Graduate School Deans Award for excellence in mentoring, and the 2010 Capers and Marion McDonald Award for Excellence in Mentoring and Advising, Pratt School of Engineering, Duke University. He served as a distinguished visitor of the IEEE Computer Society during 2005-2007 and 2010-2012, and as a distinguished lecturer of the IEEE Circuits and Systems Society during 2006-2007 and 2012-2013. He is currently an ACM distinguished speaker. He was the editor-in-chief of *IEEE Design & Test of Computers* during 2010-2012. He is currently the editor-in-chief of *ACM Journal on Emerging Technologies in Computing Systems* and *IEEE Transactions on VLSI Systems*. He is also an associate editor of the *IEEE Transactions on Computers*, *IEEE Transactions on Biomedical Circuits and Systems*, *IEEE Transactions on Multiscale Computing Systems*, and *ACM Transactions on Design Automation of Electronic Systems*. He serves on the Steering Committee of the *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, and as an editor of the *Journal of Electronic Testing: Theory and Applications (JETTA)*. In the recent past, he has served as an associate editor of the *IEEE Transactions on VLSI Systems* (2005-2009), *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2001-2013), IEEE *Transactions on Circuits and Systems I* (2005-2006), and *IEEE Transactions on Circuits and Systems II* (2010-2013). He is a fellow of the ACM, a fellow of the IEEE, and a Golden Core member of the IEEE Computer Society.