Rapid Online Fault Recovery for Cyber-physical Digital Microfluidic Biochips

Christopher Jaress, Philip Brisk
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA, USA
chrisjaress@gmail.com, philip@cs.ucr.edu

Daniel Grissom
Department of Engineering and Computer Science
Azusa Pacific University
Azusa, CA, USA
dgrissom@apu.edu

***Abstract*—Microfluidic technologies offer benefits to the biological sciences by miniaturizing and automating chemical reactions. Software-controlled laboratories-on-a-chip (LoCs) execute biological protocols (assays) specified using high-level languages. Integrated sensors and video monitoring provide a closed feedback loop between the LoC and its control software, which provide timely information about the progress of an ongoing assay and the overall health of the LoC. This paper introduces a cyber-physical control algorithm that rectifies hard and soft faults that are detected dynamically while executing an assay on a digital microfluidic biochip (DMFB), one specific LoC technology. The approach is scalable (i.e., there is no fixed limit on the number of faults that may occur), and runs efficiently in practice, thereby limiting the performance overhead incurred when a hard or soft fault occurs during assay execution.**

Keywords—Digital Microfluidic Biochip, Error Recovery

I. Introduction

A digital microfluidic biochip (DMFB) [20] is a device that manipulates discrete droplets of liquid via electrostatic actuation atop a 2-dimensional grid of electrodes. Compared to existing laboratory-on-a-chip (LoC) technologies that manipulate continuous flows of fluid, DMFBs offer three key advantages: (1) the ability to manipulate fluids individually; (2) the ability to immerse solids within liquids without the risk of clogging one or more microchannels; and (3) compatibility with a wide variety of fluid volumes. DMFB applications include DNA sequencing, immunoassays, point-of-care diagnostics, and many others [11].

Recent DMFBs integrate devices such as heaters [13], photo-detectors [14, 29], impedance sensors [22], or magnetic separators [6], which provide feedback to a PC that controls the execution of an assay (biochemical reaction) running on the DMFB, forming a feedback-control loop as shown in Fig. 1. Such a *cyber-physical* DMFB can execute assays that incorporate sensory feedback and real-time decision-making into their specification. Historically, assays were specified as directed acyclic graphs (DAGs) without decision-making or control flow. A cyber-physical DMFB can now execute assays specified as control flow graphs (CFGs), as shown in Fig. 2, which include conditions and loops whose behavior is driven by sensor feedback. Each CFG node contains a DAG, and the last operation in each DAG is either a branch or the CFG exit point, signifying that the assay has terminated.

This work was supported in part by NSF Grant CNS-1035603. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NSF.

Each control flow operation signifies a reconfiguration point, as it is not possible to predict control behavior at compile-time, and the precise configuration of droplets at the start point of each DAG is not guaranteed to be the same each time that a CFG node is invoked for execution. Consequently, it is necessary to re-compile each DAG on-the-fly as the CFG executes, i.e., the system employs a just-in-time (JIT) compiler. Each call to the compiler must schedule, place, and route the assay in real-time, i.e., the assay pauses while the compiler solves these interdependent NP-complete problems. In this context, a premium must be placed on the runtime of the JIT compiler, as opposed to solution quality.

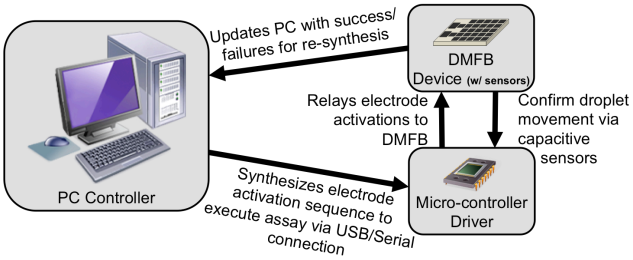


Fig. 1. A feedback-control loop for a cyber-physical DMFB with integrated capacitive-touch sensors.

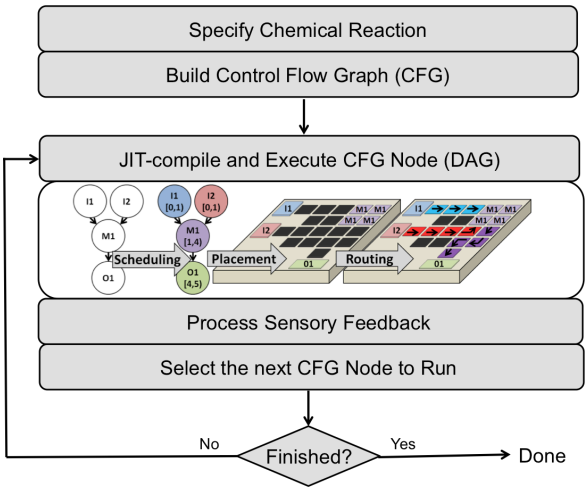


Fig. 2. Software architecture of a system that executes assays specified as control flow graphs (CFGs) in real-time.

Within the context of an online JIT compiler, this paper introduces a cyber-physical control model for DMFBs that enables recovery from faults that occur during assay execution. A hard fault refers to a device-level failure that renders a portion of the DMFB unusable. DMFBs offer abundant spatial parallelism, so a typical hard fault manifests itself as a loss of some parallelism. In the worst case, a hard fault could block the area that interfaces with an external device, such as a heater, or could block access to an I/O reservoir on the perimeter of the chip. Most hard faults are not catastrophic.

A cyber-physical DMFB can detect a hard fault in real-time by comparing the behavior of a droplet with its expected action based on a control signal sent to the device. For example, if an electrode adjacent to a droplet is activated, the expected action is droplet motion; if the droplet does not move in response, then we can assume that the droplet is stuck and the region of the chip surrounding the droplet is no longer usable. The remainder of the assay must be recompiled to avoid the faulty area. Fast re-compilation methods are needed to achieve high throughput in the presence of faults and to avoid spoilage of samples and reagents.

Soft faults, in contrast, represent erroneous assay operations that do not indicate device failure. For example, one droplet may be split into two droplets of significantly unequal volume [2, 3], or the concentration of a droplet may not be within the calibrated range of the sensor [29]. If so, the erroneous droplet must be discarded, and the part of the assay that produced the droplet is re-executed. Likewise, this entails (re-)compiling a portion of the assay to introduce new operations whose necessity could not be predicted statically.

Contribution: This paper contributes a compiler and runtime monitoring system for cyber-physical DMFBs to enable fast dynamic fault recovery. Compared to prior work, our system offers the following advantages: (1) This is the first control mechanism for cyber-physical DMFBs that handles hard and soft faults in a unified fashion; (2) the algorithm is scalable, i.e., there is no hard upper bound on the number of faults that can be tolerated; (3) the algorithm is faster than all prior scalable fault recovery algorithms that have been published to date; and (4) the general approach is intuitive and easy to implement, which favors rapid software development and a lower likelihood of errors and bug fixes later on.

II. Background and Approach

Checkpoints are automatically inserted into an assay to test for observable errors, [29]. Each checkpoint routes a droplet to a sensor/detector for assessment; if the assessment fails, an error recovery subgraph (a static program slice [27] containing all operations that may affect the droplet at the checkpoint) is inserted into the assay (Fig. 3), and the schedule, placement, and routing plan are updated. Checkpoint and error recovery subgraph insertion can be done manually or by a compiler.

The assay is initially specified as a DAG. Checkpoints and error recovery subgraphs are inserted into the assay, converting into an executable CFG, as shown in Fig. 2. Each checkpoint ends with a condition based on sensory feedback: if an error occurs, control transfers to the error recovery subgraph; if there is no error, control transfers to the next operation.

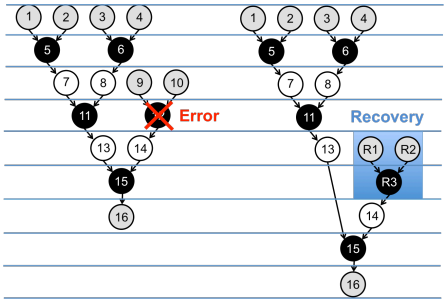


Fig. 3. A soft faults occurs at runtime in a scheduled DAG. An error recovery subgraph is introduced and the updated DAG is rescheduled.

III. Related Work

Cyber-physical integration enables real-time detection and fault recovery; several algorithms have been introduced to reschedule assay operations and reconfigure the DMFB to recover from faults. Table 1 lists the algorithms and their relevant properties in comparison to algorithms introduced in this paper. With the exception of Luo et al. [14], all of these techniques focus explicitly on hard or soft faults, but not both.

Zhao et al. [29] pioneered real-time soft fault detection and recovery for DMFBs. Their approach had two limitations: (1) all operations stop during recovery, including those that do not depend on droplets involved in the fault; and (2) operations within the error recovery subgraph must be fault-free. Subsequent work has addressed these limitations [14, 15].

Maftei et al. [16] and Alistar et al. [2] detect hard faults offline; their compiler avoids the use of faulty DMFB regions; they do not detect or recover faults that occur online.

Alistar et al. [1] and Luo et al. [15] enumerate all combinations of soft faults that might occur during assay execution, and generate all of the error recovery subgraphs that could reconfigure the system at runtime. These approaches reduce recompilation times, but can only tolerate a small number of faults due to exponentially large storage costs. They are non-scalable and cannot tolerate hard faults.

Many assays produce intermediate droplets that are not used. Early DMFB compilers dispose of all unneeded droplets, as there was no motivation to store them. Hseih et al. [10] and Luo et al. [14] optimistically store some of these intermediate droplets, as they can reduce the overhead of the fault recovery process. Our approach can support droplet re-use if desired.

Table I shows that: (1) prior work has used dynamic recompilation to recover from soft, but not hard, faults; (2) list scheduling is preferred, presumably due to its efficiency; (3) only one paper has used a polynomial-time placement heuristic, and its runtime is quadratic [14]; and (4) prior work has not considered droplet routing on recompilation.

The contribution of our work is an online recompilation technique for hard and soft faults that is scalabile, achieves a linear time complexity for placement, and accounts for droplet routing. The time router's complexity is O(MN) [21], as it uses Soukup's routing algorithm internally [24]. The average case performance of Soukup's router is less than its worst-case time complexity, and our online alternative to placement guarantees routability and helps the router converge quickly.

TABLE I. COMPARISON BETWEEN THE FAULT RECOVERY TECHNIQUES INTRODUCED IN THIS PAPER AND PRIOR WORK; THE LIMITATIONS AND WEAKNESSES OF PRIOR WORK COMPARED TO OURS ARE HIGHLIGHTED.

	Overview of Fault Detection and Recovery				During Recovery		Online Recompilation Algorithms		
Reference	Fault Type	Fault Detection	Scalable	Droplet Re-use	Assay Pauses	Tolerance to further faults	Scheduling	Placement	Routing
Zhao et al. [28]	**Soft**	Online	Yes	Yes	**Yes**	**No**	Computed Offline		
Alistar et al. [1]	**Soft**	Online	**No**	**No**	No	Yes	Enumerated Offline		**No**
Alistar et al. [3]	**Soft**	Online	Yes	**No**	No	Yes	O(nlogn)*	**No**	**No**
Maftei et al. [16]	**Hard**	**Offline**	Yes	**No**	N/A	N/A	N/A		
Luo et al. [15]	**Soft**	Online	**No**	**No**	No	Yes	Enumerated Offline		
Hsieh et al. [10]	**Soft**	Online	Yes	Yes	No	Yes	Ref. [14] or [27]		
Alistar et al. [2]	**Hard**	**Offline**	**No**	N/A	N/A	N/A	N/A		
Luo et al. [14]	Both	Online	Yes	Yes	No	Yes	O(nlogn)*	**O(MN)****	**No**
	Both	Online	Yes	Yes	No	Yes	**Iterative Improvement**		**No**
Our Work	Both	Online	Yes	Yes	No	Yes	O(nlogn)*	O(P)*** [12]	O(MN)**

* n is the number of assay operations (vertices in the DAG); list scheduling [7, 24] has an O(nlogn) time complexity.

** M and N are the DMFB length and width. O(MN) is the time complexity of Soukup's algorithm [24], used for path planning during routing [21].

*** P is the number of modules placed on the chip during reconfiguration; the O(P) time complexity is reported in ref. [12].

IV. ONLINE FAULT RECOVERY

A. Virtual Topology

The key to enable fast and efficient JIT compilation is the notion of a virtual topology [7], as shown in Fig. 4. A virtual topology segregates specific regions of the chip (work modules) to perform assay operations (mixing, dilution, storage, etc.), while leaving space (streets) between modules for droplet transport. External devices (heaters, detectors, etc.) may enhance the functionality of a work module, but do not affect droplets transported on a street.

Virtual topologies eliminate certain mistakes that arise from the interdependence between schedulers, placers, and routers. In Fig. 5(a), the schedule dictates that seven concurrent modules execute: in principle, there is enough free space on the chip to perform all operations, however, a 4x6 contiguous region cannot be found for module M7; this is an established problem called fragmentation, which occurs in dynamic placement for runtime reconfigurable FPGAs [5, 12]. In Fig. 5(b), a legal placement has been found, but the placed modules abut one another, blocking the path that droplet *D* would like to take to reach the detector on the other side of the chip. Lastly, Fig. 5(c) illustrates the complex and chaotic nature of the routing in the presence of many droplets [26].

Since the JIT compiler places a premium on runtime, the time spent to detect and correct the problems shown in Fig. 5(a) and (b) is unacceptable. The virtual topology eliminates these problems completely: the number of on-chip resources is clearly articulated to the scheduler. For example, in Fig. 4, there are four work modules that can perform mixing, splitting, and storage; one can perform heating, and another can perform detection. The scheduler has exact knowledge of what resources are available for different assay operations. As all operations occur in work modules, placement becomes a conceptually simpler binding problem [7]. Routing path blockages cannot occur, since the virtual topology ensures that all droplet routing paths (input port-to-module; module-to-module; module-to-output port) are blockage-free. Lastly, the virtual topology eliminates the chaos depicted in Fig. 5(c) due to the orderly layout of streets, and prior work [7] has demonstrated provably deadlock-free routing algorithms.

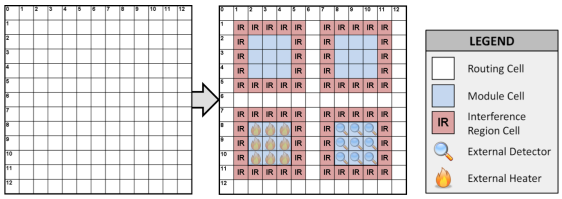
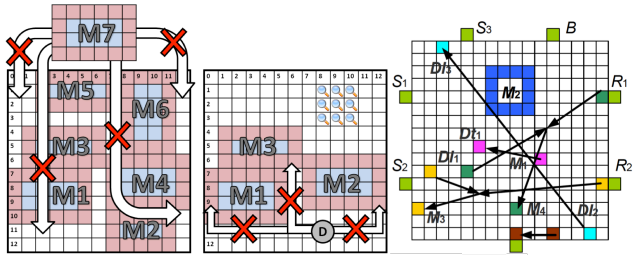


Fig. 4. Depiction of a virtual topology [7].



(a) (b) (c)

Fig. 5. Online DMFB placement is challenging because of: fragmentation (a); legal placements may have adverse affects on routing (b); and routing a large number of droplets at once can be chaotic (c) [26, Fig. 8(a)].

In a virtual topology, each work module has dedicated virtual I/O ports, which restrict the locations where droplets may enter or exit (Fig. 6). If a work module can store *k* droplets, then it requires *k* virtual input (and output) ports, along with an interference-free path from each virtual input port to its corresponding virtual output port within the module; this enables droplets to enter and leave independently without interfering with one another. The virtual I/O ports play an important role in ensuring provably deadlock-free routing at the point where droplets enter and exit work modules [7]. Droplets are allowed to wait in I/O cells as long as necessary, and spacing between them ensures that arriving droplets do not interfere with departing droplets during routing.

B. Fault Recovery Model

We assume that the assay is specified as a control flow graph with checkpoints and error recovery subgraphs inserted a-priori. We review our system's soft fault handing capabilities [8] and introduce techniques to handle hard faults.

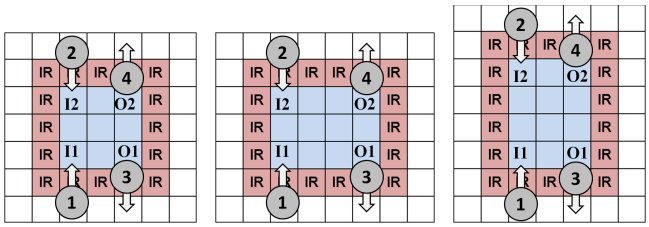


Fig. 6. Each work module in a virtual topology has dedicated virtual I/O ports where droplets can enter and leave without interfering with one another.

Soft Fault Recovery: Any erroneous droplets that are no longer usable are transported to a waste reservoir. Control flow transfers to the error recovery subgraph. The JIT compiler schedules, places, and routes the error recovery subgraph on the virtual topology using fast and efficient algorithms, e.g., list scheduling [7, 25], a binding algorithm (in lieu of placement) that selects a work module for each scheduled operation [7]; and a fast routing algorithm [7].

In contrast, Luo et al. [14] recompute the placement at each time step of the updated schedule. If the target chip is area-constrained, this approach may fail. Since Luo et al. do not include routing results, it is not possible to determine if the placements obtained by their tool are routable or not.

Hard Fault Recovery: Hard faults that occur on the DMFB surface can ruin a virtual topology. A fault in a module cell renders it unusable, while a relatively small number of faults that occur in routing cells could potentially block every pair of paths between two modules in the topology. To fix the situation, we reconfigure the virtual topology when a fault occurs to use smaller modules, as shown in Fig. 8.

The algorithm to repair the virtual topology in response to a hard fault is derived from an algorithm for online FPGA placement called Keep All Maximal Empty Rectangles (KAMER) [5]. KAMER represents the free space on the DMFB [12] using a set of overlapping maximal empty rectangles (MERs). An empty rectangle is maximal if no other rectangle encloses it. KAMER treats each work module (including its surrounding interference region [26]) as an operation; the MERs are shown in Fig. 7(a).

Assume that a hard fault affects one cell in the DMFB, as shown in Fig. 7(b). The 3x3 faulty region (FR) surrounding the cell must be avoided for the remaining lifetime of the chip. We treat the FR as a non-reconfigurable operation that persists through all future placements. Any modules that intersect the FR must be reconfigured; they are removed from the list of active modules and KAMER updates its set of MERs, as shown in Fig. 7(b).

The next step is to introduce smaller work modules with limited functionality (slower mix times [19] and reduced storage capacity), as shown in Fig. 7(c). The preferred strategy is to introduce the largest module that can fit into the reallocated space, to minimize the resulting increase in mixing time [19]. We then query the MER data structure to return the largest rectangle representing free space on the chip. If the MER is large enough to accommodate a new work module, then we add it to the chip. This repeats until no MER can accommodate any more work modules.

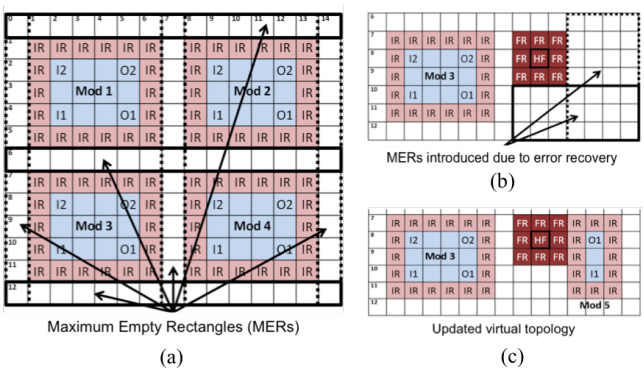


Fig. 7. On a 15x13 DMFB with 4x3 modules, (a) the MERs initially consist of the three horizontal and three vertical streets; (b) a hard fault (HF) and its surrounding faulty region (FR) makes Mod 4 unusable, resulting in two new MERs; (c) a smaller 1x3 module (Mod 5) with well-defined I/O ports is introduced and placed within the larger MER.

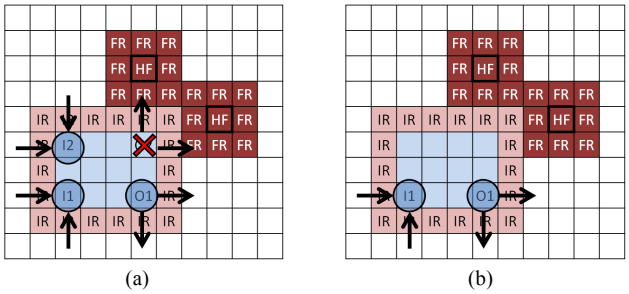


Fig. 8. Two hard faults (HFs) and their faulty regions (FRs) abut against a module. Although they will not interfere with an operation inside the module, (a) output cell O2 is unusable because there is no unobstructed path out of the module, thus (b) only one set of I/Os can be used in this module.

The last step is to determine the number of droplets that each new work module can accommodate, and select the location of the virtual I/O ports which restrict the locations where droplets may enter or exit the module (see Fig. 6).

There must be a path from an adjacent street to each virtual I/O port; otherwise, the port becomes inaccessible. In Fig. 8, a 3x4 module can store up to two droplets with two pairs of virtual I/O ports; however, two hard faults block access to virtual output port O2. As a result, the work module can only store one droplet. In principle, we could reconfigure virtual input port (I2) to be bi-directional; however, doing so would break the deadlock-free droplet routing property [7]. With less storage capacity, reducing the module size is feasible, but doing so would increase the latency of mixing operations [19].

Fig. 9 presents pseudocode for the fault recovery process.

V. Simulation Results

We implemented cyber-physical error detection and recovery in a publicly available open source compiler and simulator for digital microfluidics [8]. The experiments compare the performance and effectiveness of error detection and recovery using the virtual topology to an approach similar to the work by Luo et al. [14], which we call *free placement (FP)*. Both approaches employ list scheduling [7, 25] and a greedy droplet routing algorithm [19]. FP uses KAMER for placement [5, 12], but not for dynamic fault recovery.

```
ReconfigureFault( 3x3 Fault Region F )
//    Data structures
1.    MER data structure (includes pre-existing 3x3 fault regions): MER
2.    List of modules placed on the DMFB: L

//    Remove modules affected by the fault from the virtual topology
3.    If F occurs in a work module M or within M's interference region
          L.remove(M)
4.    ElseIf the center of F occurs in a street and F intersects the interference
      region of at least one work module
5.        Select a work module M whose interference region intersects F
6.        L.remove(M)
7.    EndIf

//    Update the MER data structure
8.    MER.insert(F)
9.    For each module M∈L
10.       MER.insert(M)
11.   EndFor

//    Introduce new, smaller work modules in the vicinity of the
//    fault, and reconstruct the virtual topology
12.   Do
13.       Boolean stop ← True
14.       For each max. empty rectangle R∈MER
15.           If R.area() >= Minimum module area
16.               M ← CreateNewWorkModule(R)
17.               MER.insert(M)
18.               stop ← False
19.           EndIf
20.       EndFor
21.   While(stop = False)
22.   L←RebuildVirtualTopology(MER)

//    Add virtual inputs and output ports of each work module
//    introduced as part of the reconfiguration process.
23.   For each newly inserted module M
24.       Select the maximum number of droplets that M can store
25.       Add virtual input and output ports for each droplet
26.   EndFor

//    Remove virtual input and output ports (if necessary) from work
//    modules that abut the faulty region F.
27.   For each module M abutting F
28.       For each virtual I/O port P blocked by F
29.           Remove P and its partner port from M
30.           Reduce the max. number of droplets that M can store
31.       EndFor
32.   EndFor
```

Fig. 9. Pseudocode describing virtual topology reconfiguration in response to a hard fault.

The virtual topology employs a restricted variation of KAMER to reconfigure itself when hard faults are detected; however, this is done once per fault discovery, and should not be confused with the usage of KAMER as a free placer that reconfigures the placement when each operation starts/stops.

We consider an exponential protein dilution assay with 5 levels (Protein-Split 5 [7]). We target a 15x19 DMFB with a 2x2 virtual topology where modules store up to two droplets. We converted the assay to a CFG [8] by inserting checkpoints and error recovery subgraphs [29]. We assume that stuck droplet faults can be detected instantaneously [4, 18]. For each experiment, we compile the assay using the virtual topology (VT) and free placement (FP) approaches. The simulator steps through the protocol at 100 Hz (10ms per cycle).

Our experiments measure the recovery time in response to hard and soft faults, and whether or not hard fault recovery is successful. All experiments were performed on a desktop PC running an Intel i7 processor clocked at 3.4 GHz with 10GB of DDR3 DRAM running Windows 8.1.

First, the assay is compiled using VT and FP; we report the initial compilation time. After building the CFG, we randomly selected 5 operations and simulated 5 soft faults; Fig. 10(a) reports the recompilation time. We then randomly selected 5 operations and simulated 5 hard faults; Fig. 10(b) reports the recompilation time. Last, we randomly generated 102 fault scenarios and recompiled the assay using both approaches; in two cases, FP failed; Fig. 10(c) reports the average recovery times for the first two faults for VT and FP for the 100 cases where both approaches were successful.

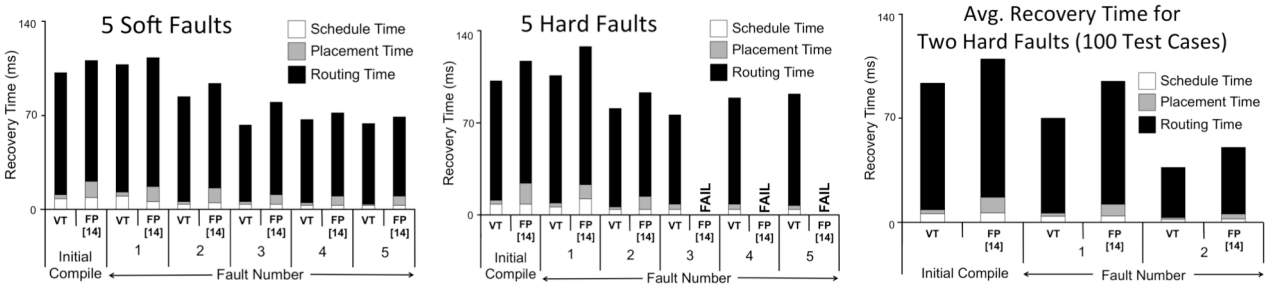
Fig. 10(a) shows three trends: (1) VT is marginally faster than FP; (2) droplet routing, not scheduling or placement, dominates recompilation time; and (3) the recovery time is to less for faults that occur later in the schedule (since more of the assay has executed, the DAG to be recompiled becomes smaller as the simulation progresses toward completion). The difference in runtime shown in Fig. 10(a) is mostly due to placement, which shows that VT's binding approach [7] is more efficient than invocation of the KAMER placer.

In Fig. 10(b), FP fails to successfully recompile the assay after the 3rd hard fault, while VT successfully recovers after all 5 faults; for the initial compilation step and dynamic recovery from the first two hard faults, the results are similar to Fig. 10(a) for soft faults. Fig. 10(c) reports similar results as well. Altogether, VT is more efficient than FP in terms of spatial resource management as hard faults are introduced into the chip. FP suffers from fragmentation, as the number of hard faults increases, while VT does not.

These results clearly indicate that dynamic recompilation could benefit from faster droplet routing algorithms whose runtimes are comparable to the scheduler and placer (VT binding and FP's invocation of KAMER); since prior work has established that droplet routing does not significantly affect total assay execution time [7, 25, 26], there is reason to believe that significant benefits could be accrued by sacrificing routing solution quality to reduce runtime.

VI. Conclusion

Online error recovery for DMFBs necessitates fast and efficient algorithms. Existing approaches do not effectively deal with the interdependence between scheduling, placement, and routing. The approach to error recovery outlined in this paper sidesteps these issues by leveraging a virtual topology: placement is converted to a binding problem, and fast, provably deadlock-free routing algorithms can be used to quickly converge. This paper has shown how to reconfigure a virtual topology in response to hard faults, thus providing graceful degradation as the chip ages. Prior work has established the viability of virtual topologies for efficient detection and recovery from soft faults.



(a) (b) (c)

Fig. 10. Recovery time and success rate using the virtual topology (VT) and free placement (FP) [14] on the Protein-Split 5 assay running on a 15x19 DMFB with (a) five soft faults; (b) five hard faults; and (c) average recovery time for 100 simulated executions of VT and FP after compiling and re-compiling the Protein-Split 5 assay with two hard faults.

REFERENCES

[1] Alistar, M., Maftei, E., Pop, P., and Madsen, J. 2010. Synthesis of biochemical applications on digital microfluidics with operation variability. In *Proceedings of the IEEE Symposium on Design Test Integration and Packaging of MEMS/MOEMS* (Seville, Spain, May 05-07, 2010) DTIP '10, 350-357.

[2] Alistar, M., Pop, P., and Madsen, J. 2013. Application-specific fault tolerant architecture synthesis for digital microfluidic biochips. In *Proceedings of the Asia and South Pacific Design Automation Conference* (Yokohama, Japan, Jan. 22-25, 2013), ASPDAC '13, 794-800. DOI= http://dx.doi.org/10.1109/ASPDAC.2013.6509697

[3] Alistar, M., Pop, P., and Madsen, J. 2012. Online synthesis for error recovery in digital microfluidic biochips with operation variability. In *Proceedings of the IEEE Symposium on Design Test Integration and Packaging of MEMS/MOEMS* (Cannes, France, April 25-27, 2012) DTIP '12, 53-58.

[4] Basu, A. S. 2013. Droplet morphometry and velocimetry (DMV): a video processing software for time-resolved label-free tracking of droplet parameters, *Lab-on-a-Chip* 13, 10 (Mar. 2013), 1892-1901. DOI=http://dx.doi.org/10.1039/C3LC50074H

[5] Bazargan, K., Kastner, R., and Sarrafzadeh, M. 2000. Fast template placement for reconfigurable computing systems. *IEEE Design & Test of Computers*, 17, 1 (Jan.-Mar. 2000) 68-83. DOI= http://dx.doi.org/10.1109/54.825678

[6] Choi, K., et al. 2013. Automated digital microfluidic platform for magnetic-particle-based immunoassays with optimization by design of experiments. *Analytical Chemistry* 85, 20 (Aug. 2013) 9638-9646. DOI= http://dx.doi.org/10.1021/ac401847x

[7] Grissom, D., and Brisk, P. Fast online synthesis of generally programmable digital microfluidic biochips. *IEEE Trans CAD* 33, 3 (Mar. 2014), 356-369. DOI= http://dx.doi.org/10.1109/TCAD.2013.2290582

[8] Grissom, D., Curtis, C., and Brisk, P. 2014. Interpreting assays with control flow on digital microfluidic biochips. *ACM Journal on Emerging Technologies in Computing Systems* 10, 3 (Apr. 2014), article #24. DOI= http://dx.doi.org/10.1145/2567669

[9] Grissom, D., O'Neal, K., Preciado, B., Patel, H., Doherty, R., Liao, N., and Brisk, P. 2012. A digital microfluidic biochip synthesis framework. In *Proceedings of the IEEE/IFIP International Conference on VLSI and System-on-a-Chip* (Santa Cruz, CA, USA, October 07 - 10, 2012). VLSI-SOC '12, 177-182, DOI= http://dx.doi.org/10.1109/VLSI-SoC.2012.6379026

[10] Hsieh, Y-L., Ho, T-Y., and Chakrabarty, K. 2012. Design methodology for sample preparation on digital microfluidic biochips. In *Proceedings of the International Conference on Computer Design* (Montreal, Canada, Sep. 30 - Oct. 3, 2012) ICCD'12, 189-194. DOI= http://dx.doi.org/10.1109/ICCD.2012.6378639

[11] Jebrail, M. J., Bartsch, M. S., and Patel, K. D. 2012. Digital microfluidics: a versatile tool for applications in chemistry, biology, and medicine. *Lab-on-a-Chip* 12, 14 (Jul. 2012), 5452-2463. DOI= http://dx.doi.org/10.1039/C2LC40318H

[12] Lu, Y. Marconi, T., Gaydadjiev, G., and Bertels, K. 2008. An efficient algorithm for free resources management on the FPGA. In *Proceedings of Design Automation and Test in Europe* (Munich, Germany, March 10-14, 2008) DATE '08, 1095-1098. DOI= http://dx.doi.org/10.1109/DATE.2008.4484923

[13] Luo, Y., Bhattacharya, B. B., Ho, T-Y., and Chakrabarty, K. Optimization of polymerase chain reaction on a cyberphysical digital microfluidic biochip. In *Proceedings of the International Conference on Computer-Aided Design* (San Jose, CA, Nov. 18-21, 2013) ICCAD'13, 622-629. DOI= http://dx.doi.org/10.1109/ICCAD.2013.6691181

[14] Luo, Y., Chakrabarty, K., and Ho, T-Y. 2013. Error recovery in cyberphysical digital microfluidic biochips. *IEEE Trans CAD* 32, 1 (Jan. 2013), 59-72. DOI= http://dx.doi.org/10.1109/TCAD.2012.2211104

[15] Luo, Y. Chakrabarty, K., and Ho, T-Y. 2012. Dictionary-based error recovery in cyberphysical digital-microfluidic biochips. In *Proceedings of the International Conference on Computer-Aided Design* (San Jose, CA, USA, November 05-08, 2012) ICCAD '12, 369-376. DOI= http://dx.doi.org/10.1145/2429384.2429463

[16] Maftei, M., Pop, P., and Madsen, J. 2013. Droplet-aware module-based synthesis for fault-tolerance digital microfluidic biochips. In *Proceedings of the IEEE Symposium on Design Test Integration and Packaging of MEMS/MOEMS* (Cannes, France, April 25-27, 2012) DTIP '12, 47-52.

[17] Mitra, D., et al. 2012. Automated path planning for washing in digital microfluidic biochips. In *Proceedings of the International Conference on Automation Science and Engineering* (Seoul, Korea, Aug. 20-24, 2012) CASE'12, 115-120. DOI= http://dx.doi.org/10.1109/CoASE.2012.6386419

[18] Murran, M. A., and Najjaran, H. 2012. Capacitance-based droplet position estimator for digital microfluidic devices. *Lab-on-a-Chip* 12, 11 (Mar. 2012) 2053-2059. DOI= http://dx.doi.org/10.1039/c2lc21241b

[19] Paik, P., Pamula, V. K., and Fair, R. B. 2003. Rapid droplet mixers for digital microfluidic systems. *Lab-on-a-Chip* 3, 4 (Nov. 2003), 253-259.

[20] Pollack, M. G., Shenderov, A. D., and Fair, R. B. 2002. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab-on-a-Chip* 2, 2 (Mar. 2002), 96-101. DOI=http://dx.doi.org/10.1039/b110474h

[21] Roy, P., Rahaman, H., and Dasgupta, P. 2010. A novel droplet routing algorithm for digital microfluidic biochips. In *Proceedings of the 20th Great Lakes Symposium on VLSI* (Providence, RI, USA, May 16 - 18, 2010) GLSVLSI '10, 441-446. DOI= http://dx.doi.org/10.1145/1785481.1785583

[22] Shih, S. C. C., et al., Digital microfluidics with impedance sensing for integrated cell culture and analysis. *Biosensors and Bioelectronics* 42, 4 (Apr. 2013) 314-320. DOI= http://dx.doi.org/10.1016/j.bios.2012.10.035

[23] Shih, S. C. C., Fobel, R., Kumar, P., and Wheeler, A. R. 2011. A feedback control system for high-fidelity digital microfluidics. *Lab-on-a-Chip* 11, 3 (Feb. 2011) 535-540. DOI= http://dx.doi.org/10.1039/C0LC00223B

[24] Soukup, J. Fast maze router. In *Proceedings of the Design Automation Conference* (Las Vegas, NV, USA, June 19-21, 1978) DAC '78, 100-102, DOI= http://dx.doi.org/10.1109/DAC.1978.1585154

[25] Su, F., and Chakrabarty, K. 2008. High-level synthesis of digital microfluidic biochips. *ACM Journal on Emerging Technologies in Computing Systems* 3, 4 (Jan. 2008), article #16. DOI= http://dx.doi.org/10.1145/1324177.1324178

[26] Su, F., Hwang, W., and Chakrabarty, K. 2006. Droplet routing in the synthesis of digital microfluidic biochips. In *Proceedings of Design Automation and Test in Europe* (Munich, Germany, March 06-10, 2006) DATE '06, 1-6. DOI= http://dx.doi.org/10.1109/DATE.2006.244177

[27] Weiser, M. 1984. Program slicing. *IEEE Trans. Software Engineering* 10, 4 (July 1984) 352-357. DOI= http://dx.doi.org/10.1109/TSE.1984.5010248

[28] Xu, T., and Chakrabarty, K. 2007. Functional testing of digital microfluidic biochips In *Proceedings of the IEEE International Test Conference* (Santa Clara, CA, USA, Oct. 21-26, 2007) ITC'07. DOI= http://dx.doi.org/10.1109/TEST.2007.4437614

[29] Zhao, Y., Xu, T., and Chakrabarty, K. 2010. Integrated control-path design and error recovery in the synthesis of digital microfluidic biochips. *ACM Journal on Emerging Technologies in Computing Systems* DOI= http://dx.doi.org/10.1145/1777401.1777404