Efficient Parallel Testing and Diagnosis of Digital Microfluidic Biochips

SIDDHARTHA DATTA, BHARAT JOSHI, ARUN RAVINDRAN,
and ARINDAM MUKHERJEE
University of North Carolina at Charlotte

Microfluidics-based biochips consist of microfluidic arrays on rigid substrates through which movement of fluids is tightly controlled to facilitate biological reactions. Biochips are soon expected to revolutionize biosensing, clinical diagnostics, environmental monitoring, and drug discovery. Critical to the deployment of the biochips in such diverse areas is the dependability of these systems. Thus robust testing and diagnosis techniques are required to ensure adequate level of system dependability. Due to the underlying mixed technology and mixed energy domains, such biochips exhibit unique failure mechanisms and defects. In this article efficient parallel testing and diagnosis algorithms are presented that can detect and locate single as well as multiple faults in a microfluidic array without flooding the array, a problem that has hampered realistic implementation of several existing strategies. The fault diagnosis algorithms are well suited for built-in self-test that could drastically reduce the operating cost of microfluidic biochip. Also, the proposed alogirthms can be used both for testing and fault diagnosis during field operation as well as increasing yield during the manufacturing phase of the biochip. Furthermore, these algorithms can be applied to both online and offline testing and diagnosis. Analytical results suggest that these strategies that can be used to design highly dependable biochip systems.

Categories and Subject Descriptors: B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; I.6.5 [**Simulation and Modeling**]: Model Development; I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis

General Terms: Algorithms, Reliability, Performance

Additional Key Words and Phrases: Microfluidic biochip, multiple faults, testing, fault tolerance, defect tolerance, droplet flooding, reconfigurability, microfluidics

ACM Reference Format:

Datta, S., Joshi, B., Ravindran, A., and Mukherjee, A. 2009. Efficient parallel testing and diagnosis of digital microfluidic biochips. ACM J. Emerg. Technol. Comput. Syst. 5, 2, Article 10 (July 2009), 17 pages. DOI = 10.1145/1543438.1543443 http://doi.acm.org/10.1145/1543438.1543443

Authors' address: S. Datta, B. Joshi (contact author), A. Ravindran, A. Mukherjee, Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223-0001; email: bsjoshi@uncc.edu.

DOI 10.1145/1543438.1543443 http://doi.acm.org/10.1145/1543438.1543443

1. INTRODUCTION

Over the past decade, research in integrated circuit testing has broadened from digital test to include the testing of analog and mixed-signal devices. More recently, new test techniques for mixed-technology microelectromechanical systems (MEMS) are also receiving attention [Deb et al. 2000, 2004; Dhayni et al. 2004]. As MEMS rapidly evolve from single components to highly integrated systems for safety-critical applications, dependability is emerging as an important performance parameter. Fabrication techniques such as silicon micromachining lead to new types of manufacturing defects in MEMS [Deb et al. 2000]. Moreover, due to their underlying mixed technology and multiple energy domains, such composite microsystems exhibit failure mechanisms that are significantly different from those in electronic circuits. In fact, the 2003 International Technology Roadmap for Semiconductors (ITRS)[1] recognizes the need for new test methods for disruptive device technologies that underlie composite microsystems, and highlights it as one of the five difficult test challenges beyond 2009.

Microfluidics-based biochips constitute an emerging category of mixed-technology microsystems [Kerkhoff 1999; Kerkhoff and Hendriks 2001; Kerkhoff and Agar 2003]. Recent advances in microfluidics technology have led to the design and implementation of miniaturized devices for various biochemical applications. These microsystems, referred to interchangeably in the literature as microfluidics-based biochips, lab-on-a-chip and bioMEMS [Pollack et al. 2002; Su et al. 2004], promise to revolutionize biosensing, clinical diagnostics and drug discovery. Such applications can benefit from the small size of biochips, the use of microliter/nanoliter sample volumes, lower cost, and higher sensitivity compared to conventional laboratory methods.

The earlier generations of microfluidics-based biochips were based on the manipulation of continuous liquid flow through fabricated microchannels [Mir et al. 2000]. Liquid flow was achieved either by external pressure sources, integrated mechanical micropumps, or by electrokinetic mechanisms such as electro-osmosis. Recently, a novel microfluidics technology has been developed to manipulate liquids as discrete microliter/nanoliter droplets. Following the analogy of digital electronics, this technology is referred to as “digital microfluidics” [Srinivasan et al. 2004]. Compared to continuous-flow systems, digital microfluidics offers the advantage of dynamic reconfigurability and architectural scalability.

The level of system integration and the complexity of digital microfluidics-based biochips are expected to increase in the near future due to the growing need for multiple and concurrent bioassays on a chip [Su et al. 2003]. However, shrinking processes, new materials, and the underlying multiple energy domains will make these biochips more susceptible to manufacturing defects. Moreover, some manufacturing defects are expected to be latent, and they may manifest themselves during field operation of the biochips. In addition, harsh operational environments may introduce physical defects such as particle

[1] http://public.itrs.net/Files/2003ITRS/Home2003.htm.

contamination during field operation. Consequently, effective test and diagnosis techniques are required to ensure system dependability as biochips are deployed for safety-critical applications such as field diagnostics tools to monitor infectious disease, decode genes, and biosensors to detect biochemical toxins and other pathogens. Here testing implies a procedure to detect whether something has gone wrong (fault detection) while diagnosis implies a procedure to determine exactly where the fault is located (fault location). In this article efficient and effective parallel algorithms are presented that can detect and locate single as well as multiple faults, avoid flooding, and pinpoint the actual fault as opposed to faulty regions that are typically identified by the strategies proposed in the literature. Furthermore, they render themselves well for built-in self-test thereby eliminating the need of having elaborate and expensive external test equipment, and can be used to test the biochips both during the manufacturing phase as well as field operation.

The organization of the remainder of the article is as follows. Related prior work is described in Section 2. While the fault modeling of digital microfluidic biochips and fault types are discussed in Section 3, Section 4 describes the proposed methodologies. Finally, conclusions are drawn in Section 5.

2. PRIOR WORK

Although research in the design of digital microfluidics-based biochips has gained considerable momentum in recent years [Pollack et al. 2002; Srinivasan et al. 2004; Su et al. 2004], to the best of our knowledge only a few researchers have reported work on the testing of digital microfluidic biochips [Davids et al. 2006; Su et al. 2004; 2003; 2004; 2004; 2005; 2005; 2006; 2006; 2006; 2005]. A cost-effective test methodology for digital microfluidic systems was first described in Srinivasan et al. [2004]. Physical defects in such systems were analyzed and faults were classified as being either catastrophic or parametric. In all the proposed strategies faults are detected by electrically controlling and tracking the motion of test droplets. An optimal test planning method for the detection of catastrophic faults in digital microfluidic arrays was investigated in [Su et al. 2004; 2004]. It is based on a graph model of the array and a problem formulation based on Hamiltonian paths in a graph. An efficient concurrent testing method that interleaves test application with a set of bioassays was first proposed in Su et al. [2005]. Reconfiguration and defect tolerance techniques for biochips were described in Davids et al. [2006] and Su et al. [2006].

Prior work on the testing of digital microfluidics-based biochips is based on simplistic assumptions regarding the impact of certain defects on droplet flow. For example, a common defect seen in fabricated microfluidic arrays is a short circuit between two adjacent electrodes [Pollack et al. 2006]. It was assumed in Davids et al. Su et al. [2004; 2003; 2004; 2004; 2005; 2005; 2006; 2006; 2006; 2005] that this defect causes a droplet to be stuck at one of the two electrodes irrespective of the orientation of liquid flow. Based on this, Hamiltonian paths are used to detect catastrophic faults in microfluidic arrays [Su et al. 2004; 2005]. One of the problems with this approach is that although finding Hamiltonian paths in grid structures is well known, checking the existence of Hamiltonian path in a given graph is NP-complete. Thus it would be expensive to determine such paths in the microfluidic array after it is reconfigured based on the results of the fault diagnosis.

Furthermore, an attempt was made to experimentally validate the above assumption. Experiments suggested that the effect of such a short-circuit defect on droplet flow depends on whether the droplet flow path is perpendicular to the two shorted electrodes or aligned with them [Su et al. 2006]. A test procedure for such defects should therefore, not only test single cells as in [Srinivasan et al. 2004; Su et al. 2004; 2005] using Hamitonian paths, but it should also focus on pairs of cells and the traversal of droplets from one cell to all its neighbors. A different testing methodology based on Euler path was developed to detect catastrophic faults, including those caused by electrode shorts [Su et al. 2005]. Although this method is better than the one based on Hamiltonian paths in terms of diagnosibility, it has one major shortcoming. When an electrode *i*is faulty, then this fault cannot be distinguished from any other fault that may exist in a closed tour starting at *i*in the Euler path.

Moreover, most of the methods in the literature can diagnose only single fault and when the droplet fails to reach the sink (indicating the presence of a fault) these methods do not retrieve the droplet back to the source. This could eventually lead to flooding, an undesirable phenomenon in which droplets accumulate at an electrode. These issues were recently addressed in [Davids et al. 2006]. However, sequential algorithm is used for testing whose time complexity is $O(n^2)$. Recently a parallel test and diagnosis strategy was proposed [Xu et al. 2007]. While the strategy has interesting features the fault coverage is not high and there are possibilities for misdiagnosis and flooding.

In this article, integrated testing and diagnosis methodologies are presented in which several parts of the biochip are tested simultaneously to reduce the test time while offering the advantages of the strategy of Davids et al. [2006]. The degree of parallelism is bounded above by the number of sources available.

3. FAULT MODELING

The microfluidic biochip discussed in Su et al. [2006] is based on the maneuvering of microliter-nanoliter droplets using the principle of electrowetting-on-dielectric (EWOD) [Pollack et al. 2002]. Detailed descriptions of different kinds of faults that may occur in a microfluidic chip are given in Srinivasan et al. [2004] and Su et al. [2006]. In this article it is assumed that the biochip uses the principle of EWOD and the types of faults that could occur will be those identified in Srinivasan et al. [2004] and Su et al. [2006]. These faults include:

(1) Short between the droplet and the electrode. This typically happens due to the application of unusually high voltage resulting in the dielectric breakdown. Under this condition the droplet typically undergoes electrolysis and further movement of the droplet is not possible.

(2) Electrode short due to metal connection between two adjacent electrodes due to abnormal metal layer deposition and etch variation during fabrication. Under this fault condition a droplet is stuck between the two shorted electrodes and further movement of the droplet cannot be achieved.

(3) Open circuit between the electrode and the control source due to variability in the manufacturing process. An electrode cannot be activated under this fault condition.

(4) Several conditions, such as particle contamination, cause the impedance between the plates to become high leading to fluidic open. A droplet cannot move across the obstacle under this fault condition.

Thus it can be seen that an effective testing strategy should test each electrode and edge in the biochip. It is assumed that no failure takes place during any round of testing.

4. TESTING AND DIAGNOSIS STRATEGIES

The testing algorithm, which is based on divide-and-conquer strategy, exploits the sources to test several parts of the biochip concurrently. Initially the grid is partitioned into subgrids so that each subgrid is accessible by a source/sink. Each source/sink dispenses test droplets to at most two subgrids. Thus the number of subgrids is not more than twice the number of sources available. Each subgrid is represented by a unique identifier $[x, y]$, the size of every subgrid is represented as (m, n) each electrode position is represented as {i, j}, and a source/sink is represented by <i, j>.

4.1 Partitioning Strategy

The partitioning algorithm is based on two observations: (i) recognizing that each subgrid should be accessible by a source for the purpose of testing the partitioning strategy should ensure that all the subgrids are on the periphery of the biochip; (ii) the smallest size of a subgrid that could be used is (3, 3) since a smaller grid could result in a misdiagnosis.

Let the total grid size be $N \times M$ while the size of each subgrid size be (m, n). The strategy assigns at most two subgrids to each source. Thus the total number of subgrids $\leq$ 2(number of sources). Among the four boundaries of the biochip if k (≤ 4) of them has sources then for the purpose of partitioning the total number of cases that have to be considered are given by:

$$\sum_{k}^{4} = 1\binom{4}{k} = 2^4 - 1 = 15. \tag{1}$$

Please note that $k = 0$ is not considered since it implies that a droplet cannot be introduced into the biochip and such a chip is not realistic. After carefully evaluating the trade-offs between the optimum number of partitions given the number and distribution of the sources, the efficiency of the partitioning algorithm, and ease with which concurrency can be exploited during the testing phase (Section 4.2) it became apparent that three basic partition configurations should be considered and each of the 15 cases could be mapped on to one of these configurations. The first configuration is trivial, in which the grid cannot be partitioned. In the second configuration the subgrids are created only in one dimension (Figure 1(a)). In the third configuration the grid is divided into

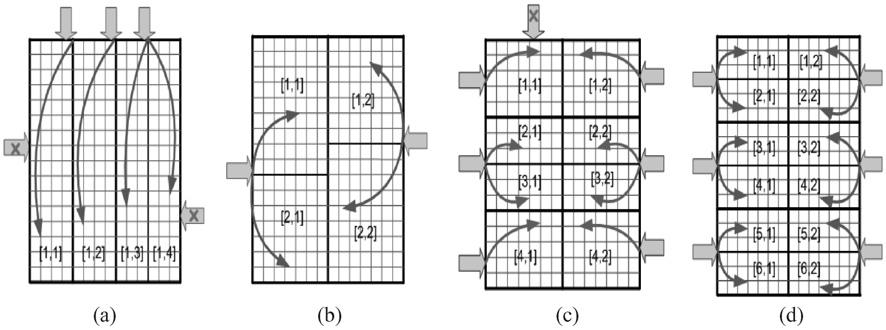


Fig. 1. Examples where k = 1, 2, and 3. Sources not used for testing are marked by X.

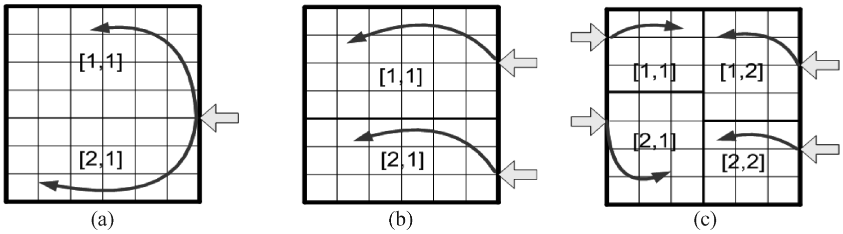


Fig. 2. Examples where k = 1 and 2.

two halves in one dimension and each half is further divided into subgrids in the other dimension (Figure 1(d)).

When there is only source available then the grid cannot be divided into subgrids. The grid is divided into subgrids in only one dimension for $k = 1$, and the number of sources is greater than one (Figure 1(a)). Two basic groups are created when $k = 2$ based on the relative positions of the two boundaries where the sources are present. If the boundaries are orthogonal then the one with more sources is selected and the subgrids are identified using the strategy for $k = 1$. This is a greedy step since the boundary with more sources is selected. If both the boundaries are selected then care has to be taken to ensure that each subgrid that is created is accessible by a source. A better strategy is to consider each boundary separately and the one that gives more subgrids should be selected. But this will increase the run time of the algorithm. The other group (when $k = 2$) consists of sources on opposite boundaries (Figure 1(d)).

When $k = 3$ it implies that two boundaries opposite to each other and one of the remaining two boundaries have sources in which case the two boundaries opposite to each other are used to identify the subgrids. The grid is divided into two halves in one dimension and each half is further divided into subgrids in the other dimension based on the sources as shown in the Figure 1(c). The sources available at the third boundary are not used to create the subgrids and thus they will not be used by the testing algorithm. When all the four sides have sources, then the two sides opposite to each other with more sources are selected. Once again, this is a greedy step, and better solution is possible at the cost of increasing the run time of the partitioning algorithm as mentioned for the case when $k = 2$. The algorithm is presented in the following.

Algorithm 1. Partitioning

```
if (num_sources = 1) then
{
   sub-grids = 1;
   if (sub-grid size $\geq$ (m, n)) then
   the grid is testable;
}
else
{
   Case 1: All the sources are on the left or right (top or bottom) boundary of the grid
   {starting from top (left) enqueue sources;
   merged_electrodes = 0;
   while (queue is not empty)
   {
   dequeue source <I, j> /* the source may be adjacent two sources <a, b> and <c, d> or
                                          adjacent to a boundary and one source */
   if (source is adjacent to a boundary and a source <a, b>)
        Boundary_Source;
   else if (the source is adjacent two sources)
        Source_Source;
end while
}
end Case 1
}

 {
Case 2: At least one pair of opposite boundaries of the grid has sources
   {
   Divide the grid into two halves midway between the selected pair of boundaries with
   sources;  /* This will become one of the four boundaries of each of the subgrids */
                 /* for each boundary starting from top (left) enqueue sources;*/
    merged_electrodes = 0;
      while (queue is not empty)
      {
      dequeue source <I, j> /* the source may be adjacent two sources <a, b> and <c, d>
                                          or adjacent to a boundary and one source */
        if (source is adjacent to a boundary and a source <a, b>)
            Boundary_Source;
        else if (the source is adjacent two sources)
            Source_Source;
       end while
       }
      end Case 2
      }
      end if_else
      }
      Boundary_Source
      {
         temp_electrodes = number_of_electrodes between the source and the boundary;
         if ((temp_electrodes $\geq$ 3) and (merged_electrodes + $\lfloor |i - a|/2 \rfloor \geq 3$) (or
     $\lfloor |i - b|/2 \rfloor \geq 3$))
         create two sub-grids – between the source and the boundary, and between the
         source and midway between the two sources;
 else if ((temp_electrodes + merged_electrodes +$\lfloor |i - a|/2 \rfloor$ (or $\lfloor |i - b|/2 \rfloor$) $\geq 3$)
{
 create one grid consisting of (temp_electrodes + merged_electrodes +$\lfloor |i - a|/2 \rfloor$)
 rows ($\lfloor |i - b|/2 \rfloor$ columns);
        merged_electrodes = 0;
}
else
   merged_electrodes = temp_electrodes + merged_electrodes +$\lfloor |i - a|/2 \rfloor$
   ($\lfloor |i - b|/2 \rfloor$); /* source cannot be used for testing. Testing should be done by the
                        next (previous if it is the last source) source */
end Boundary_Source
}
Source_Source
{
                              /* Let the source be adjacent two sources <a, b> and <c, d> */
if ((merged_electrodes +($\lfloor |i - a|/2 \rfloor$ $s \geq 3$) $and$ ($\lfloor |i - c|/2 \rfloor \geq 3$)  ($or$
(merged_electrodes + $\lfloor |i - b|/2 \rfloor s \geq 3$) and ($\lfloor |i - d|/2 \rfloor \geq 3$))
       { create two sub-grids –create two sub-grids between the source and midway
between the two sources;}
 else if ((merged_electrodes +$\lfloor |i - a|/2 \rfloor$ +$\lfloor |i - c|/2 \rfloor$) $\geq 3$)
create one grid consisting of (merged_electrodes + $\lfloor |i - a|/2 \rfloor$ +$\lfloor |i - c|/2 \rfloor$)  rows
(columns);
         else
merged_electrodes = (merged_electrodes +$\lfloor |i - a|/2 \rfloor$+ $\lfloor |i - c|/2 \rfloor$ $/ * source$
 cannot be used for testing */
end Source_Source}
```

4.2 Testing Strategy

The testing and diagnostic technique described in Davids et al. [2006] is modified to detect and locate faults in each subgrid. According to Su et al. [2005], it is assumed that the maximum capacity of each electrode is to accommodate not more than 2 droplets at any point of time avoiding the possibility of flooding. The basic technique proposed in this article exploits parallelism at two levels, by running the algorithm concurrently on a subset of the subgrids and within each subgrid, columns (rows) are tested in parallel. Ideally all the subgrids should be tested concurrently, but testing two adjacent subgrids concurrently may potentially lead to misdiagnosis since they may share some electrodes or their electrodes may be adjacent to each other. Thus a realistic solution should partition the subgrids into different sets such that the subgrids in each set can be tested concurrently. It should be noted that maximal concurrency can be achieved by creating minimal number of subsets.

A graph theoretic approach is used for the purpose of identifying the subsets. An undirected graph G, which is an ordered pair $G(V, E)$ where V is a set nodes and E is a set of edges, is constructed such that V is the set of all the subgrids and an edge exists between two nodes if the corresponding subgrids share at least one electrode or their electrodes are adjacent. Node coloring, which is the assignment of colors to the nodes so that no two adjacent nodes share the same color, is used to identify the subgrids that can be tested concurrently. The aim is

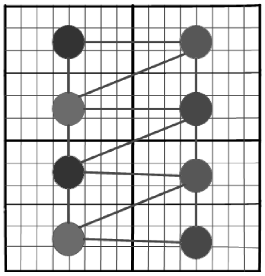
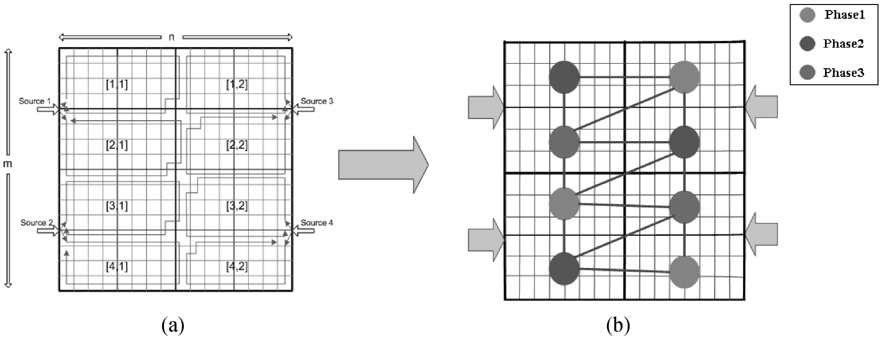


Fig. 3. Subgrids and node coloring of the corresponding graph with four colors.



(a) (b)

Fig. 4. An example of a typical system and its corresponding graph.

to use the least number of colors (chromatic number of the graph) to maximize the concurrency. It is well known that the problem of finding the chromatic number of a graph is NP-hard and the corresponding decision problem (i.e., is there a coloring which uses at most k colors?) is NP-complete. However, the nodes of a grid structure, such as the graphs of some of the subgrids generated by the partitioning strategy of Section 4.1, can be colored by at most four colors (Figure 3). Nodes in the first column (row) can be colored by two colors and the nodes in the second column (row) can be colored by another two colors. This implies that all the subgrids can be tested in at most four parallel steps.

The number of colors used can be reduced to three by ensuring that the subgrids [i, j] and [i+1, j+1] (Figure 4(a)) do not share any electrode and they do not have electrodes that are adjacent to each other. Thus the corresponding nodes in the graph will not be adjacent. This is illustrated in Figure 4(b). Figure 5 shows different configurations of the subgrids and the corresponding graphs with node coloring.

The test strategy consists of the following three basic steps:

Step 1: Outer Loop Test. In the first step the outer loop of each subgrid is tested. A test droplet is dispensed into the subgrid from the droplet source. It is routed through the subgrid traversing all the electrodes and their respective edges of the outer loop. Finally the droplet returns to the source/sink. Failure of the droplet to return to the sink indicates a fault in the outer loop.

The algorithm when implemented on the partitioned grid cannot test the edges connecting adjacent subgrids. Thus the algorithm is modified such that

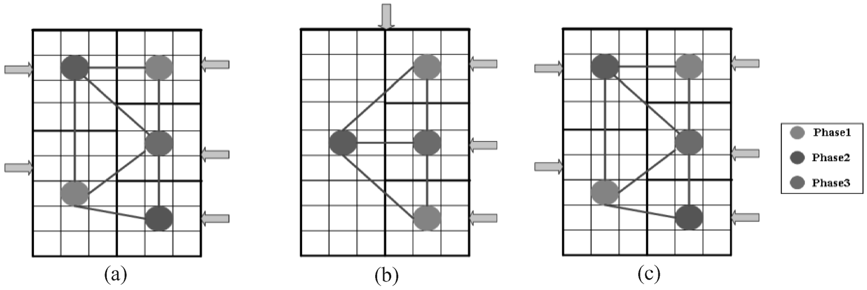


Fig. 5. Subgrids and node coloring of the corresponding graphs.

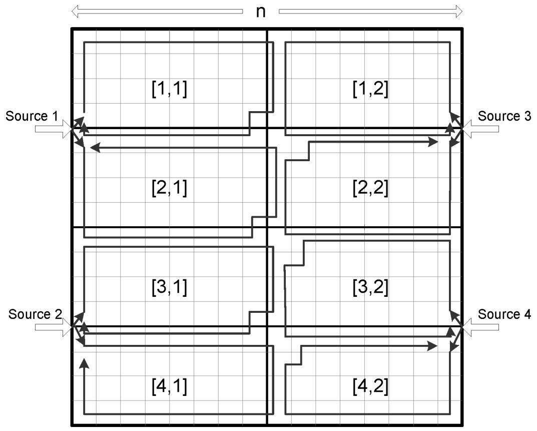


Fig. 6. Example illustrating the paths of test droplets through their respective subgrids.

the test droplets traverse the outer edges of its adjacent subgrids resulting in the adjacent subgrids test droplets' paths to overlap. Referring to Figure 6, the outer and inner horizontal paths of each test droplet of subgrid [1, 1] is changed to traverse through the left most column of [1, 2] and the uppermost row of [2, 1]. Similarly, the test droplets programmed to traverse the inner vertical paths of [1, 1] are modified to traverse through the uppermost row of [2, 1]. Thus the path length of these test droplets increase by the number of extra electrodes to be traversed.

Three phases are required to test the outer loops of the subgrids shown in Figure 6. This is illustrated in Figure 4. The inner loops traverse through parts of the outer loops. Testing of the outer loops of the sub grids results in increase in latency of the testing strategy, but avoids the possibility of flooding [Davids et al. 2006] and misdiagnosis. Figure 5 shows examples of various configurations of the subgrids all requiring three phases to test the outer loops.

Step 2: Internal Vertical Path Test. The internal vertical paths are tested in three iterations. In the first iteration all the odd numbered paths, except the first one, of all the subgrids are tested in parallel. The test droplets are routed from various sources such that they are at the beginning of all the odd-numbered vertical paths. This is followed by routing the droplets through the

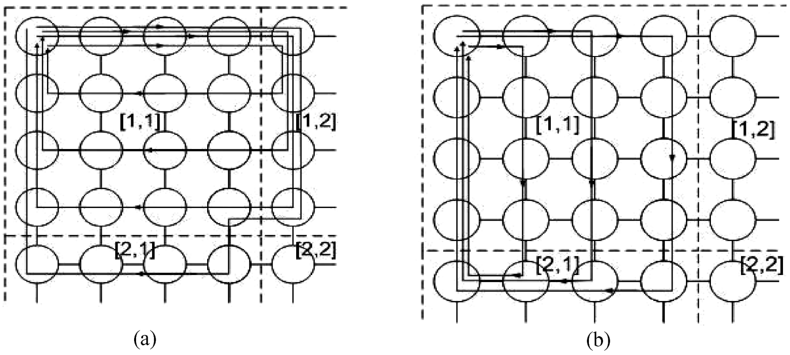


Fig. 7. Testing paths of inner vertical and horizontal paths of the subgrid [1, 1].

odd-numbered vertical paths in parallel. If a droplet is not retrieved at the sink indicates a fault in the corresponding vertical path. Assuming that the test droplets are routed in the clockwise direction (Figure 7(b)), the droplets that are stuck in vertical paths due to faults are retrieved by applying voltages in the counterclockwise direction. In the last iteration the first inner loop is tested. However, the droplet is routed in the counterclockwise direction so that it passes through the outer loop (which is already tested) first. This is to ensure that any fault in the inner vertical path is detectable.

Step 3: Internal Horizontal Path Test. The inner horizontal paths are tested in the similar way (Figure 7(a)), except that the first inner horizontal path is not tested separately.

4.3 Analysis of the Testing Strategy

Any single or multiple faults present in the array that does not allow movement of a droplet can be detected since the corresponding outer-loop or vertical and horizontal paths tests of the subgrid in which the fault is located will fail. Thus the fault detection coverage, which is a measure of the strategy's ability to detect faults, is 100% for the type of faults that inhibit the movement of droplets.

It can be seen that the time taken to traverse the outer paths of all the subgrids is bounded above by

$$T_O(n, m) = 2(4n + 4m) \tag{2}$$

This includes the time to retrieve the droplet if a fault is present in the outer loop. The time taken to traverse all the inner rows of all the subgrids is the time required to route the droplet through the longest loop which is bounded above by

$$T_{IH}(n, m) = 2(4n + 4m) \tag{3}$$

Similarly, the time taken to traverse all the inner columns of all the subgrids is bounded above by

$$T_{IH}(n, m) = 2(4n + 4m) \tag{4}$$

Thus the whole array can be tested in $O(n + m)$.

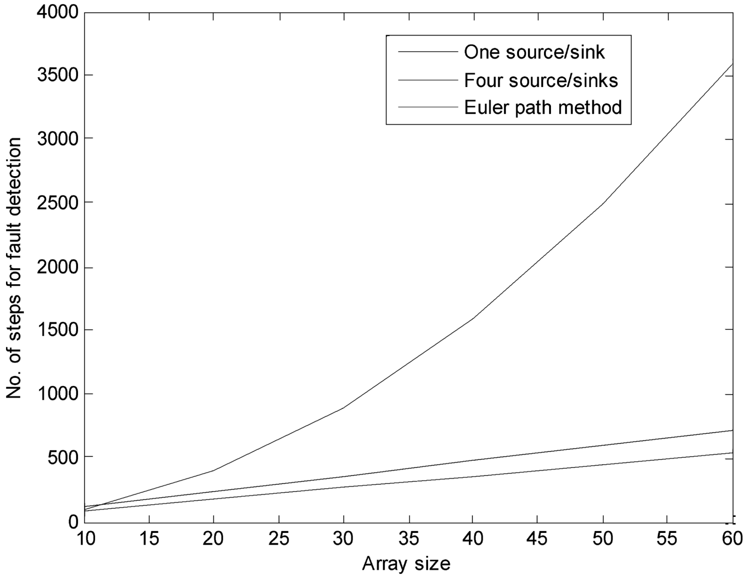


Fig. 8. Comparison of testing complexity of the proposed and the Euler-circuit method.

Comparison of the proposed method and the Euler-path method was performed based on the simulation of the offline option on arrays that varied from 10 × 10 to 60 × 60 electrodes. Analytical and simulation results (Figure 8) are consistent suggesting that while the testing time increases quadratically for the Euler-path method it increases linearly for the proposed method. Simulation results were obtained for randomly injected single faults. Also, the testing time decreases for the proposed method as the number of source/sink increases with the limiting case that the size of a subgrid cannot be less than (3, 3) as discussed in Subsection 4.1. Finally the testing time is independent on the location of the defect/fault. It can be seen that while the strategy is efficient some boundary electrodes will be activated quite frequently that may eventually decrease their dependability.

4.4 Fault Diagnosis

If a test droplet is not retrieved at the sink as scheduled, a fault is diagnosed in its traversal path. An attempt is made to retrieve the droplet by applying voltages to the electrodes in a counterclockwise direction starting from midpoint of the loop. If the droplet is retrieved, then the fault is present in that half of the loop. Otherwise, it is present in the other half. The segment of the loop where the droplet is present is further divided into two halves and an attempt is made to retrieve the droplet in a counterclockwise direction, as done before. This search, based on the binary search technique, is done until the droplet is retrieved at the sink. At this point, the smallest segment containing the fault has been identified. A droplet is sent through the source in a clockwise direction until the end of the segment in which the fault is present by applying voltages to appropriate electrodes sequentially. Once again, an attempt is made to retrieve the droplet by applying voltage anti clockwise. The immediate clockwise neighbor and the edge connecting these two nodes are now considered to be suspicious. Therefore, it is necessary to conduct further diagnosis to identify the fault. A droplet is sent in a counterclockwise direction to the immediate counterclockwise neighbor of the node that is currently considered to be suspicious. An attempt is now made to retrieve the droplet in the clockwise direction by applying voltages to electrodes sequentially, starting from the second neighbor of the suspicious node in a clockwise direction. If the droplet is retrieved successfully, it indicates that the droplet was stuck at the immediate neighbor of the suspicious node in the clockwise direction, implying that the suspicious node is faulty while the edge is still considered to be suspicious. Further tests to identify the status of the suspicious edge are not done, since it is incident to the faulty node and edges incident to the faulty node will no longer be used. If the droplet is not retrieved, a second attempt is made by applying voltages to the electrodes sequentially starting from the immediate neighbor of the suspicious node. Successful retrieval of the droplet implies that the suspicious node is fault-free, while the suspicious edge is faulty.

4.5 Analysis of the Diagnosis Strategy

It can be seen that any single fault can always be located by the binary search technique. In the worst case the single fault could be associated with the last electrode in the loop, that is, the one just before the sink. In this case the total number of searches that need to be performed is $(2n + 2m)$. During each search the number of time steps is bounded above by $O(n + m)$. Thus the run time of the binary search is in $O((n + m)log(n + m))$.

Similarly a class of multiple faults where only a single fault is present in any vertical or horizontal loop of a subgrid can be located correctly. All these faults can be located by applying the binary search technique to all the vertical loops simultaneously. This is followed by applying the search technique to the horizontal loops. If the subgrid sizes are sufficiently small then larger class of multiple faults can be correctly located. The run time of the diagnosis algorithm to detect multiple faults is in $O((n + m)log(n + m))$ since the binary search will require up to $log(2n + 2m)$ steps and the number of steps required to retrieve the droplet in each search step is bounded above by $O(n + m)$.

Conclusions similar to the testing methods can be drawn for diagnosis (Figure 9) except that the diagnosis time is dependent on the fault location for the proposed method.

If any row or a column has multiple faults then only the fault that has blocked the test droplet can be located. The probability that a fault cannot be located under such a scenario is now calculated. Let p_f be the probability that an electrode fails. All the failures are assumed to be independent. Also, assume that the test droplets were routed in the loops in the clockwise direction during the testing phase. If the fault is located at electrode {j, k} with respect to its subgrid's source/sink, then the probability that it cannot be located is given by:

$$p_f(1 - (1 - p_f)^{(j-1)})(1 - (1 - p_f)^{(m-k)}) \tag{5}$$

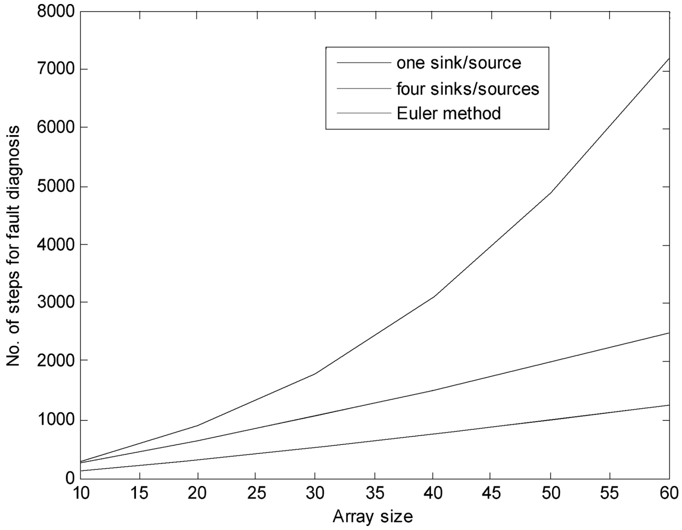


Fig. 9. Comparison of diagnosis complexity of the proposed and the Euler-circuit method.

for a subgrid of size (n, m). From Equation (1) it is clear that the probability that a fault cannot be located is higher when its location is farther away from the source/sink node in the subgrid. While such a situation of a fault not being diagnosed arises within a subgrid, multiple faults within a row or a column of the complete array can be completely diagnosed when each column or a row of each subgrid has at most one fault.

4.6 Online Testing

Online testing implies that the fault testing of the system runs concurrently with the optimized execution of various bioassays. The bioassays and the test droplets are dispensed simultaneously from their respective allocated sources and routed finally to the sinks. The bioassays are dispensed to execute according to a scheduling and optimization algorithm, such as the one proposed in Su et al. [2006]. If a fault is detected, the current assay droplets operating in the faulty region are *flushed* (sent back to the reservoirs or sinks). If the current assay's previous macro step was executed in the faulty region, the droplets are simply sent to the sinks and are regarded useless. If not, they are directed to the reservoirs. The function ***Flush( )*** in the algorithm serves this purpose. After all the current bioassays are flushed or sent to the reservoirs, the fault location process begins. After locating the fault, the system reads through all the assays that have previously passed through the faulty region. The ***Reconfigure_mixer_loc( )*** checks and if required, reallocates mixer positions to avoid any fault regions. The ***Reconfigure( )*** function assures that the bioassay droplets follow a fault free path. Re-scheduling for the previously affected macro steps is done. These assays are put up as high priority as compared to the bioassays which have yet to be dispensed by the scheduler, so that as soon as there is free grid space available these assay droplets are dispensed from

Algorithm 2. Online Testing

```
Simulated_annealing  loop
 {
Input (Total grid size of the bio-chip, number of sources/sinks available to dispense
test droplets, number of sources available to dispense bio - assay droplets, bio - assay
dependency graph)
Partition
Test and Diagnosis
/* faults are located. Now the optimization begins*/
   Division of the bio - assay dependency graph into macro steps.
     For (each macro step) loop
        {
Resource_allocation (number of sources available to dispense bio – assay droplets,
number of input droplets of current macro step)
        Calculation of location of mixers.
        If (Fault located)
           Previous macro steps are checked.
        If (previous macro step allocated droplets path passes through fault located)
         {
        While (all affected macros steps not flushed)
                    {
                 Flush (Present macro step or stuck droplet from any previous macro step).
                   Current_macro step = macro step-x.  /* x is the number of previous
executed macro steps*/
                 }
}
         Mixer location is checked for faulty electrodes.
         If (faulty electrode/electrode is detected) then
         Reconfigure_mixer_loc (mixer location, fault location/locations)
            Calculation of the 3 shortest paths between the source and mixer location for
each input droplet of the current macro step.
         PRNG(Pseudo Random Number Generator) randomly selects any of the
3 paths as the critical path.
         Critical paths ($\Delta_D$) of all droplets are checked for faulty electrodes.
         If (faulty electrode/electrode is detected in any droplet's path) then
         Reconfigure (fault location/locations, droplet path)
         Scheduling (All bio - assay droplets running in current macro step, all test
                              droplets running, time)
         Total time (of that simulated _annealing iteration) = Total time (of that simulated
       _annealing iteration) + time
         /*time taken to complete all operations of current macro step*/
} end for
} end Simulated_annaeling loop
```

their scheduled sources. The pseudocode for the online testing algorithm is given by Algorithm 2.

Figure 10 shows snapshots of a macro step of the dispensed bioassays with clustered and dispersed faults. The figure gives a graphical representation of the reconfigured paths and locations of bio-assay droplets and mixers respectively. The complete simulation consisted of three bio assays, each having multiple macro steps.

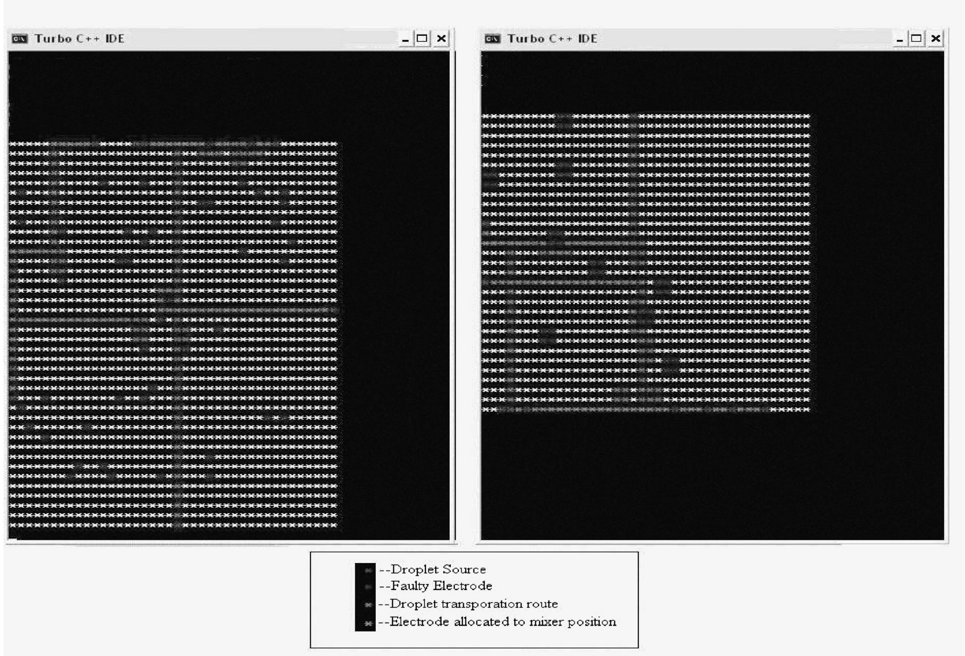

Turbo C++ IDE
Turbo C++ IDE
--Droplet Source
--Faulty Electrode
--Droplet transporation route
--Electrode allocated to mixer position

Fig. 10. A Simulation result of a macro step the assay demonstrating the reconfiguration of droplets' paths around the detected dispersed and clustered faults.

5. CONCLUSIONS

Efficient testing and diagnosis methodology for digital microfluidic biochips is proposed. It is based on a partitioning technique that increases parallelism in testing and diagnosis by exploiting the available sources and sinks. Single as well as multiple faults can be detected and located. Analytical and simulation results suggest that the proposed method is efficient and shows significant improvement over existing work. The method renders itself well for online testing by using any existing optimization algorithm. While the online testing results in less latency as compared to offline testing it requires some overhead for scheduling the bio-assays.

The proposed testing and diagnosis strategies will not work when the fault is at the source. Under such a scenario the partitioning algorithm can be run again without considering that source. The strategies will work as far as there is at least one source is available.

REFERENCES

Davids, D., Datta, S., Mukherjee, A., Joshi, B., and Ravindran, A. 2006. Multiple fault diagnosis in digital microfluidic biochips. *ACM J. Emerg. Tech. Comput. Syst.*, *2*, 4, 262–276.

Deb, N. and Blanton, R. 2000. Analysis of failure sources in surface-micromachined MEMS. In *Proceedings of the IEEE International Test Conference*. 739–749.

Deb, N. and Blanton, R. 2004. Multi-modal built-in self-test for symmetric microsystems. In *Proceedings of the IEEE VLSI Test Symposium*. 139–147.

Dhayni, A., Mir, S., and Rufer, L. 2004. MEMS built-in-self-test using MLS. In *Proceedings of the IEEE European Test Symposium*. 66–71.

Kerkhoff, H. 1999. Testing philosophy behind the micro-analysis system. In *Proceedings of SPIE: Design, Test and Microfabrication of MEMS and MOEMS.* Lecture Notes in Computer Science, vol. 3680. 78–83.

Kerkhoff, H. and Hendriks, H. 2001. Fault modeling and fault simulation in mixed micro-fluidic microelectronic systems. *J. Electron. Test. Theor. Applic. 17*, 427–437.

Kerkhoff, H. and Acar, M. 2003. Testable design and testing of micro-electro-fluidic arrays. In *Proceedings of the IEEE VLSI Test symposium*. 403–409.

Mir, S., Charlot, B., and Courtois, B. 2000. Extending fault-based testing to microelectromechanical systems, *J. Electron. Test. Theor. Applic. 16*, 279–288.

Pollack, M., Shenderov, A., and Fair, R. 2002. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip 2*, 96–101.

Srinivasan, V., Pamula, V., and Fair, R. 2004. An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids, *Lab on a Chip*. 310–315.

Su, F. and Chakrabarty, K. 2004. Architectural-level synthesis of digital microfluidics-based biochips. In *Proceedings of the IEEE International Conference on CAD*. 223–228.

Su, F., Ozev, S., and Chakrabarty, K. 2003. Testing of droplet-based microelectrofluidic systems. In *Proceedings of the IEEE International Test Conference*. 1192–1200.

Su, F., Ozev, S., and Chakrabarty, K. 2004. Test planning and test resource optimization for droplet-based microfluidic systems. In *Proceedings of the IEEE European Test Symposium*. 72–77.

Su, F., Ozev, S., and Chakrabarty, K. 2004. Concurrent testing of droplet-based microfluidic systems for multiplexed biomedical assays. In *Proceedings of the IEEE International Test Conference*. 883–892.

Su, F., Chakrabarty, K., and Pamula, V. K. 2005. Yield enhancement of digital microfluidics-based biochips using space redundancy and local reconfiguration. In *Proceedings of the DATE Conference*. 1196–1201.

Su, F., Chakrabarty, K., and Pamula, V. K. 2005. Defect tolerance for gracefully-degradable microfluidics-based biochips. In *Proceedings of the IEEE VLSI Test Symposium*. 321–326.

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

Su, F., Ozev, S., and Chakrabarty, K. 2006. Concurrent testing of digital microfluidics-based biochips. *ACM Trans. Des. Automat. Electron. Syst. 11*, 2, 442–464.

Su, F. and Chakrabarty, K. 2006. Yield enhancement of reconfigurable microfluidics-based biochips using interstitial redundancy. *ACM J. Emerg. Technol. Comput. Syst. 2*, 2, 104–128.

Su, F., Hwang, W., Mukherjee, A., and Chakrabarty, K. 2005. Defect-oriented testing and diagnosis of digital microfluidics-based biochips. In *Proceedings of the IEEE International Test Conference*. 21.2.

Received January 2008; revised May 2008, July 2008; accepted July 2008 by Krishnendu Chakrabarty