Priority Scheduling in Digital Microfluidics-Based Biochips*

Andrew J. Ricketts, Kevin Irick, N. Vijaykrishnan, Mary Jane Irwin
The Pennsylvania State University, University Park, PA, 16802
{rickets,irick,vijay,mji}@cse.psu.edu

Abstract

Discrete droplet digital microfluidics-based biochips face problems similar to that in other VLSI CAD systems, but with new constraints and interrelations. We focus on one such problem of resource constrained scheduling for digital microfluidic biochips. Since the problem is NP-complete, finding the optimal solution is a very time expensive task. We propose a hybrid priority scheduling algorithm solution directly applicable to digital microfluidics with the potential to yield near optimal schedules in the general case in a very short time. Furthermore we propose the use of configurable detectors that allow for even more improved system performance.

1. Introduction

Microfluidics-based biochips, or lab-on-a-chip, provide a platform that performs various chemical reactions on the hundreds of nanoliter scale [1] saving reagent expenses and the quantity of donor sample required to perform laboratory examinations. These devices dispense reagents and samples, mix these together, and then detect the result of the reaction. In comparison to traditional laboratories these can be fully automated and require small fractions of reactants to complete analysis. The International Technology Semiconductor Roadmap for 2004 expects that by 2010 that biological device integration will become a challenge [2].

The two main architectural implementations of microfluidics are digital and continuous flow microfluidics. Digital microfluidics based biochips controls discrete droplets through the entire process from dispensing to moving and mixing to detecting and discarding, but continuous flow systems use fixed channels to flow the fluid. Digital microfluidic systems provide a more scalable architecture that is easier to fabricate when compared to continuous flow systems [3, 4]. Furthermore, the reconfigurable nature of digital microfluidic motion and mixing offers other benefits since the path taken by a droplet at any given time is determined by an external microcontroller that can individually access any location in the entire biochip offering the potential use of fault detection and avoidance techniques to route around faults [5]. The microcontroller can be reprogrammed later if a better schedule is found or even update itself after detecting a fault. These capabilities do not exist in the fixed system of continuous flow microfluidics. Movement and reaction between discrete droplets have been demonstrated in a digital microfluidic biochip [1, 3, 6]. The term digital versus continuous stems from the analogy between binary and analog systems as in digital microfluidics the drops are either present at a given location or not. Figure 1 shows a conceptual diagram of a digital microfluidic system.

An important use of these systems would be a low cost detector for the most common metabolic disorder in the world, diabetes [6]. Other metabolites detection have been demonstrated [3, 6] that can be used for determining the presence of certain physiological disorders.

Design tools in digital microfluidics initially focused on device-level physical modeling of single components, but to build large-scale systems top-down system level design tools are required. Although some work has been done to fill this gap overall the number of tools is still relatively small [7]. To extend the available sources of such tools we propose a hybrid priority scheduling

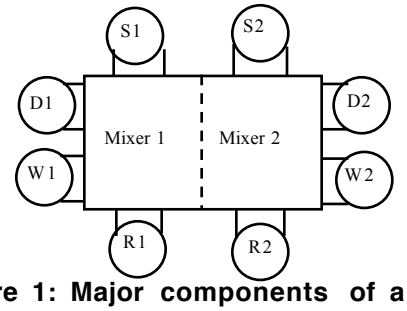

Figure 1: Major components of a digital microfluidic- based biochip Sx/Rx – sample/reagent dispenser, Dx – detector for reagent x, Wx – available waste reservoir

* This work was supported in part by NSF 0093085 and NSF 0303981 grants

algorithm, HPA, to perform the scheduling allocation in a resource-constrained digital microfluidic biochip that is required to perform concurrent biomedical assays. The tests in [7] have shown that while direct application of a genetic algorithm, GA, can provide results that approach the lower bound solution, the time required for achieving this solution is long. The HPA solution proposed in this work reduces the run times and also provides better solutions than the GA by adding to the initial population solutions that seek to minimize resource conflicts.

Section 2 discusses related work. Section 3 reviews the principle of digital microfluidics and formulates the scheduling problem. Section 4 explains the algorithmic implementation. Section 5 explains the improved detector proposed and compares its scheduling results with previously published results. Finally, Section 6 draws conclusions.

2. Related work

An ad-hoc priority based genetic algorithm, GA, was proposed in [7] to schedule resource allocation in a resource-constrained digital microfluidic biochip. It was shown to provide better results than a modified list scheduling (M-LS) algorithm that considered the urgency value of a node in the topologically sorted graph based on dependency constraints. A weakness of the genetic algorithm in [7] was that it took a comparatively long time to finish, as long as five times longer, and its results were only slightly better than the M-LS.

It can be noted that, in part, this longer running time was due to the unnecessarily large search space given to the GA. For a system with *Sm* samples and *Rn* reagents the genetic encoding used in [7] results in a search space that has *(4(Sm*Rn))!* possible solutions. The proposed HPA reduces this search space to only *Sm*Rn*. In addition to the HPA we use a hybrid genetic algorithm, HGA, with knowledge-augmented operators [8] to improve the performance of the GA by initializing part of the first population using a heuristic with clear priority rules minimizing the resource conflicts.

3. Principles of digital microfluidics with problem formulation

3.1 Dispensing:

The assay begins with droplet dispensing, which introduces the droplet into the array. Each droplet type is given $Nr \geq 1$ reservoirs which act as dispensers. The time for dispensing is mainly based on system parameters independent of droplet properties [9]. This suggests equal dispensing time which is assumed to be one time unit in our experiments. Each sample needs to be assayed with each reagent requiring *Rn*Sm* samples to be dispensed. Similarly each reagent needs to be present for each of these samples requiring a total of *Sm*Rn* reagents to be dispensed.

3.2 Mixing/storage:

Droplet mixing and storage are interrelated as each node in the array may store a droplet at a particular instant and then later be reused in conjunction with other nearby nodes as part of a mixer, but it may only perform one of these two functions at a time. The maximum number of mixers, *Nmixer*, is related to the number of available storage units, *Nmemory*, by the ratio approximation, i.e.

$$Nmixer = ratio*Nmemory. \quad (1)$$

In our experiments, this ratio is set to 0.25 such that each mixer consumes four storage locations. This causes a constant tradeoff between storage space and available mixers limiting the ability to store all droplets or mix all droplets in parallel.

The time taken for each mixture, for a given type of mixer, is dependent on the viscosities of the droplets. Since the reagents are highly diluted by the same fluid before dispensing, usually water, they have similar viscosities [3] causing the mixing time to largely depend on the viscosity of the sample. Consequently, samples such as saliva, plasma, and serum have mixing times independent of the diluted reagents. Based on experimental results from [3], we assume the mixing time for plasma is 5, serum is 3, urine is 4, and saliva is 6 (dispensing time units). The total number of mixing steps that need to be completed is *Sm*Rn,* one for each sample/reagent pair.

3.3 Detection/disposal:

After a sample and the appropriate reagent have been mixed the resulting drop may need to be stored or if the appropriate detector is available then it may proceed to the detection stage. The detection units are initially considered reagent specific requiring at least one detector per reagent, $Nd \geq 1$. A total of Sm*Rn detections need to be performed, one for each sample/reagent mixed product. In our experiments, the detection time for glucose, lactate, pyruvate and glutamate are assumed to be 5, 4, 6 and 5 time units respectively [3]. Finally, after detection it is assumed

that the drop is discarded into the waste reservoir. Extending this base system to increase the possibilities of parallelism and reconfigurability of the system we propose a detector improvement in Section 5 comparing its effect with the base system.

3.4 The complete sequencing graph model:

With droplet speeds of 20 cm/s [7] and electrode pitch of 1.5 mm [3], the worst case conflict free movement time in a 25 X 25 array will only be 0.36 seconds compared with the 2 seconds for dispensing and 6 seconds for the slowest mixing time. Furthermore, the majority of droplet movements are for much smaller distances as compared to traversing the worst case paths in the array. Consequently, droplet movement time is considered negligible in our simulations similar to that assumed in [7].

The complete sequencing graph summarizing the various steps involved in a biochip assay is illustrated in Figure 2. A total of *2(Sm*Rn)* samples and reagents are dispensed. The droplets are combined by performing a total of *Sm*Rn* mixing operations and are followed by *Sm*Rn* detections. No-operations are placed at the beginning and end of the sequence to operate as source and sink nodes. Each node in the graph is assigned a weight corresponding to the time required for the operation assigned to the node. The edges are communication costs between nodes representing droplet movement time and are given a weight of zero.

4. Solution to the scheduling problem

We based our solution on a GA formulation to search the possible scheduling solutions. The operation of a GA is based on the natural process of survival of the fittest and genetic evolution. A GA encodes the problem using a chromosome and evaluates the fitness of each chromosome based on the desired objective function. A large number of randomly generated chromosomes form the initial population. New generations are created by selecting the fittest chromosomes from prior generations and from those created by genetic modifications of chromosomes from the prior generation. We refer the reader to [8] for more details

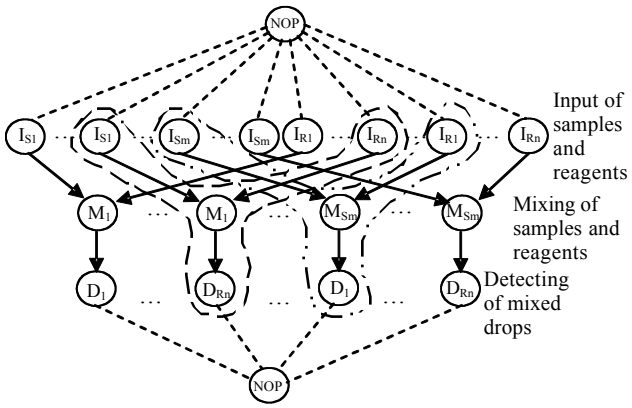

Figure 2: The complete sequencing graph model of a multiplexed biomedical assay (Adapted from [7])

on genetic algorithms and focus on the specific problem implemented as a GA.

Encoding: Each chromosome has a total of *4(Sm*Rn)* entries as shown in Figure 3. Each entry within these four groups of *Sm*Rn* genes is assigned a number $0 \le p <$ *Sm*Rn* representing the priority of that operation with respect to competing operations in the same group. A competing operation is one that requires the same resource. For example, only *Nr* (number of reservoirs for a given droplet type) droplets of sample *Si* can be dispensed at a given time step so every dispensing operation of sample *Si* competes with each other directly. By extension dispensing operations also compete with other samples and reagent dispensing as the mixer/storage limitations must be fulfilled before a dispensing operation is allowed to proceed. As such a reagent dispensing operation with a higher priority than a sample dispensing operation may block the dispensing of the sample due to storage limitations.

Previously [7] assigned random number priorities in the range $1 \le p \le 4*Sm*Rn$ to each operation. Our improvement is based on realizing that some of the operations do not directly conflict with each other and encoding the priorities for the sample dispensing, reagent dispensing, mixing, and detection is performed independently. For example, sample and reagent dispensing do not directly compete with detection priorities and can be assigned priorities independent of each other requiring only

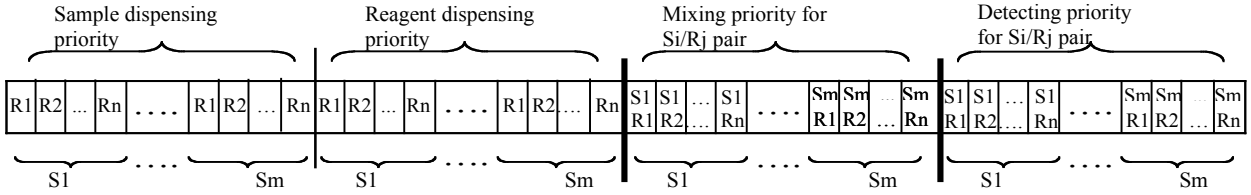

Figure 3: The encoded chromosome

Table 1: An example chromosome priority encoding with 2 samples and 2 reagents

	Priority															
	Sample Dispensing				Reagent Dispensing				Mixing				Detecting			
k	S1		S2		R1	R2	R1	R2	S1/R1	S1/R2	S2/R1	S2/R2	S1/R1	S1/R2	S2/R1	S2/R2
1	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
2	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0
3	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
4	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2

*Sm*Rm* priorities, one for each possible relative priority within each group. The major drawback of the *4*Sm*Rm* priority scale is the existence of a large number of encoded chromosomes that will yield the same solution with no improvement. Consider an encoding with priorities as follows: *1,2,3,…q,k,4*Sm*Rm*. This allows all values of *k* in the range $q+1 \leq k < 4*Sm*Rm$ to yield the same solution wasting significant cycles without any improvement in the actual solution.

Holding resources without making use of them while preventing others that could use these resources from gaining access have a detrimental effect on the overall scheduling time. We allocate the same priorities to all operations that are directly interdependent to remove the possibility of hold and wait due to unfulfilled incoming edge requirements. Consider the two sets of highlighted operations in Figure 2, each of the operations in group *i* are given the same priority. Similarly, those in group *j* are

```
Initialization: Generate initial population P(0) with the first
SmRn following equations (2) & (3)
min fitness = ∞
while(min fitness is too large)
    for each chromosome of P(t)
        for each cycle
            for all detectors
                if detection is finished
                    empty detector
                if detector is empty and appropriate mixed drop is ready
                    assign highest priority mixed drop to detector
            if mixed drop is ready for an empty appropriate detector
                assign highest priority mixed drop to detector
            else if appropriate detector is not available
                for all completed mixed drops
                    store mixed drop
            if mixer is available
                dispense next highest priority sample and reagent
        fitness = time to complete latest finishing task
        if fitness < min fitness
            min fitness = fitness
            min schedule = schedule
    P(t+1) = Reproduction(P(t))
    P(t+1) = Crossover(P(t))
    P(t+1) = Mutation(P(t))
    P(t) = P(t+1)
Print min schedule
Print min fitness
```

Figure 5: Pseudo code for scheduling algorithm

also assigned the same priority, but a different one to that of group *i*. This unique priority assignment removes any resource conflicts as resources are always reserved for the highest priority operation with its incoming edges satisfied.

In the first chromosome genes dependent on *SiRj* are given the priority:

$$^{1}p_{SiRj} = (Si-1)*Si + Rj - 1 \qquad (2),$$

and in the *kth* chromosome, with $k \leq Sm*Rn$, the priority is:

$$^{k}p_{SiRj} = ((^{1}p_{SiRj} + (k-1))\%(Sm*Rn)) \qquad (3).$$

Simply stated these cycles through all the priorities giving each interdependent operation group an opportunity to be all the available priorities. While all the *Sm*Rn!* combinations are not tried our experiments have shown that this is sufficient to achieve near optimal solutions especially as the problem sizes grows. An example of the chromosomes that would result from these priority assignments is shown in Table 1.

Fitness Function: The finishing time of the schedule determined from the priorities assigned in the chromosome is used as the fitness function. We also consider the storage requirements along with the priorities in determining the

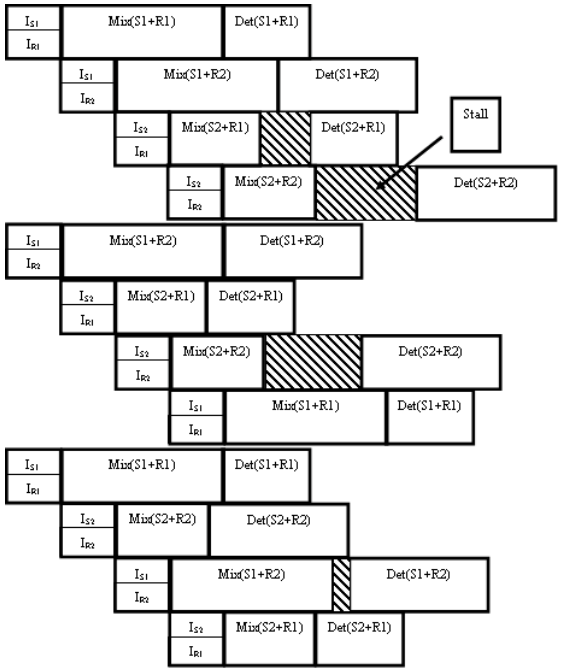

Figure 6: Rearranging operations to shorten schedule for a two sample, two reagent system that can support up to three concurrent mixers and one detector for each reagent type

finishing time. For example, if no vacancy exists in the mixer, then a droplet dispensing may be delayed beyond that determined by its priority in the encoding.

Additionally we use a HGA formulation that uses domain specific knowledge in the initializing of the chromosomes of the first generation instead of random initializing. Our HGA utilizes the heuristic that if all resources required for a specific operation between *Ri* and *Sj* incur no stalls then the overall completion time can be reduced. Hence, our HGA solution initializes the priorities within the four different types of operations in the chromosome to provide the same relative priority for all operations involving the same reagent/sample pair. For example, if *S1* and *R2* need to be mixed, dispensing, mixing, and detection of *S1* and *R2* are assigned the same priority. All pairs are assigned unique priorities for that chromosome. We use a sliding approach to assign priorities based on our heuristic to initialize S*mRn* chromosomes in the initial population. The termination condition was to meet a specified bound, usually within 10% of the lower bound, for the schedule completion time. Figure 5 contains the pseudo code of the HGA implementation. Figure 6 shows the rearranging of scheduled operations that, due to limiting over allocation of detection resources, lowers the finial completion time.

5. Influence of tunable detectors

As was stated earlier the detectors are reagent specific. The detectors are tuned to observe change specific for each analyte. As such previous works have suggested the use of light wave specific light sources near the peak absorption

Table 2: Results for Hybrid Genetic Algorithm (1 scheduling time unit (tu) = 2 seconds)

Example	Weights for mix operations	Weights for detection operations	LB [7] (tu)	M-LS [7] (tu)	GA [7] (tu)	HGA (tu)	HGA2 (tu)	Real CPU Time (sec)
(Nr=Nd=1 Nmixer=3) Sm=2, Rm=2	S1=5,S2=3	D1=5,D2=4	15	17	15	16	15	0.8
(Nr=Nd=1 Nmixer=4) Sm=2, Rm=3	S1=5,S2=3	D1=5,D2=4 D3=6	17	19	17	18	17	1.2
(Nr=Nd=1 Nmixer=5) Sm=3, Rm=3	S1=5,S2=3 S3=4	D1=5,D2=4 D3=6	23	26	25	23	22	2.1
(Nr=Nd=1 Nmixer=7) Sm=3, Rm=4	S1=5,S2=3 S3=4	D1=5,D2=4 D3=6,D4=6	23	27	26	23	23	2.7
(Nr=Nd=1 Nmixer=9) Sm=4, Rm=4	S1=5,S2=3 S3=4,S4=6	D1=5,D2=4 D3=6,D4=6	29	35	34	29	29	3.9
(Nr=Nd=1 Nmixer=9) Sm=4, Rm=5	S1=5,S2=3 S3=4,S4=6	D1=5,D2=4 D3=6,D4=6 D5=8	37	N/A	N/A	38	34	10.1
(Nr=Nd=1 Nmixer=9) Sm=5, Rm=5	S1=5,S2=3 S3=4,S4=6 S5=3	D1=5,D2=4 D3=6,D4=6 D5=7	39	N/A	N/A	44	38	32.7
(Nr=Nd=1 Nmixer=9) Sm=5, Rm=6	S1=5,S2=3 S3=4,S4=6 S5=3	D1=5,D2=4 D3=6,D4=6 D5=7,D6=6	39	N/A	N/A	44	39	60.3
(Nr=Nd=1 Nmixer=12) Sm=5, Rm=7	S1=5,S2=3 S3=4,S4=6 S5=3	D1=5,D2=4 D3=6,D4=6 D5=8,D6=6 D7=7	45	N/A	N/A	45	44	75.8
(Nr=Nd=1 Nmixer=15) Sm=7, Rm=7	S1=5,S2=3 S3=4,S4=6 S5=3,S6=7 S7=6	D1=5,D2=4 D3=6,D4=6 D5=8,D6=6 D7=7	61	N/A	N/A	61	59	177.6

wavelengths [6]. The requirement for specific mixed drops to be observed under a particular detector imposes a sequential bottleneck. As such we propose the use of white light sources, LEDs, in conjunction with reconfigurable detectors. A wavelength detector placed on the opposite end as shown in Figure 7 following a similar design to that of

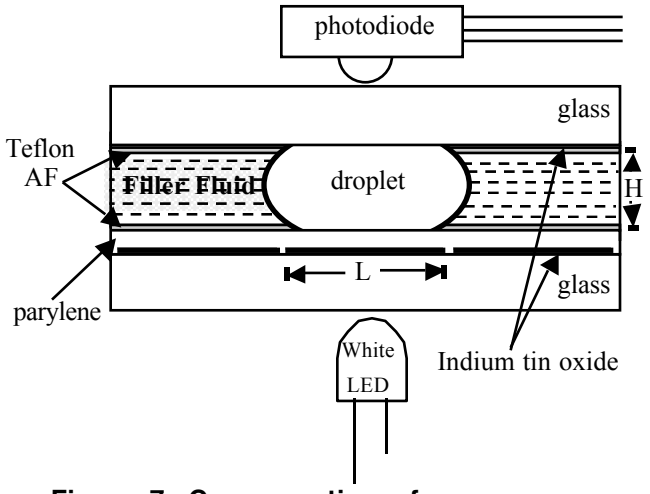

Figure 7: Cross section of a reconfigurable digital microfluidic-based biochip detector (Adapted from [6])

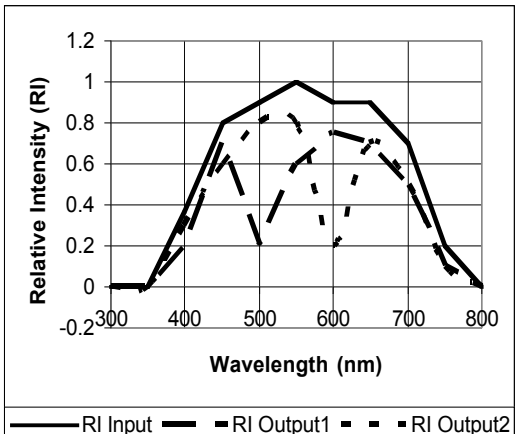

Figure 8: An example of the expected input/output light wavelengths intensity from the reconfigurable detector

[6], but offers the potential to be more flexible as a white light source will contain all the wavelengths in the visible spectrum. The expected input and output light spectrum when the system is prototyped is shown in Figure 8. In this work the tunable detector design is only considered conceptually to demonstrate its utility in reducing schedule length. There are various issues in the design of the detector such as system calibration, effects of environmental conditions on the system, repeatability and accuracy of the measurements that still need to be demonstrated. In this work the measurement is expected to be performed by selectively observing only the wavelengths under interest.

The availability of this system was assumed and added into the priority scheduler changing the appropriate detector references to simply be any detector. It was assumed that detection under this system would take similar times as previously published results based on reagent specific detectors. Table 2 shows the comparison of schedule finish times obtained using our HGA with the normal detection system (HGA) and with the improved detector system (HGA2). The real CPU time for HGA2 is included in the last column of the table. The implementation details of HGA2 and HGA are virtually identical so their running times are roughly equal. These were compared with the lower bound (LB), GA, and M-LS from [7]. The results for M-LS and GA were available only for the first five tests. The second five were attempted to explore that the system was able to quickly reach near optimal solutions for even larger problem sizes. We find that both HGA and HGA2 works well compared to the LB, specifically for larger problem sizes. They outperform M-LS for all cases and the GA for the three largest problems. Of particular interest is the ability of HGA2 to surpass the previous LB as there is an increased opportunity for parallelism in the detection stage. The LB is calculated using the worse case of all detections occurring sequentially. By allowing longer detections to proceed in parallel using the proposed tunable detector offers the possibility to overcome this sequential bottleneck. These simulations were run on a Sun Blade 1000 750MHz UltraSPARC-III CPU. The longest running schedule for the first five examples took under 4 seconds real CPU time to complete for HGA and HGA2 compared with 5 minutes for M-LS and 25 minutes for GA from [7].

6. Conclusion

We presented a hybrid genetic algorithm for scheduling operations for bio-assays in a microfluidic bio-chip based on priority scheduling. We have demonstrated that the domain specific heuristic that is used to initialize the GA population and our new encoding scheme that reduces the effective search space help to provide schedules with shorter completion times and shorten the algorithm runtimes in finding the solutions as compared to prior work. We have determined a scalable algorithm that can determine near optimal digital microfluidic operation schedules. Furthermore, we proposed an improved detector which could allow the system to exceed its previous lower bound. This detector system seems best able to outperform the normal detector system mainly when some detection times are significantly longer than mixing times. Consequently, our future work will focus on the implementation issues of the tunable detector in detail.

References

1. M.G. Pollack, R.B. Fair, and A.D. Shenderov, “Electrowetting-based actuation of liquid droplets for microfluidic applications”, Applied Physics Letters, vol. 77, pp. 1725-1726, 2000.

2. International Technology Roadmap for Semiconductors (ITRS), http://www.itrs.net/Common/2004Update/2004_10_AP.pdf

3. V. Srinivasan, V.K. Pamula, M.G. Pollack and R.B. Fair, “Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears on a digital microfluidic platform”, Proc. μTAS, pp. 1287-1290, 2003

4. M.G. Pollack, A. D. Shenderov and R. B. Fair, “Electrowetting-based actuation of droplets for integrated microfluidics”, Lab on a Chip, vol. 2, pp 96-101, 2002.

5. F. Su and K. Chakrabarty, "Defect tolerance for gracefully-degradable microfluidics-based biochips", in press.

6. V. Srinivasan, V.K. Pamula, M.G. Pollack and R.B. Fair, “A digital microfluidic biosensor for multianalyte detection”, Proc. IEEE Conf. MEMS, pp. 327-330, 2003.

7. F. Su and K. Chakrabarty, "Architectural-Level Synthesis of Digital Microfluidics-Based Biochips", Proc. IEEE International Conference on CAD, pp. 223-228, 2004.

8. D. E. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning (1989), Addison-Wesley, Massachusetts.

9. H. Ren and R. B. Fair, “Micro/Nano Liter Droplet Formation and Dispensing by Capacitance Metering and Electorwetting Actuation”, Technical Digest IEEE-NANO, pp. 36-38, 2002