Behavioral Modeling and Performance Evaluation of Microelectrofluidics-Based PCR Systems Using SystemC

Tianhao Zhang, Krishnendu Chakrabarty, *Senior Member, IEEE*, and Richard B. Fair, *Fellow, IEEE*

***Abstract*—Composite microsystems that incorporate microelectromechanical and microelectrofluidic devices are emerging as the next generation of system-on-a-chip (SOC). We present a performance comparison between two types of microelectrofluidic systems (MEFS): continuous-flow systems and droplet-based systems. The comparison is based on a specific microelectrofluidic application—a polymerase chain reaction (PCR) system. The behavioral modeling, simulation, and performance evaluation are based on a SystemC design environment. The performance comparison includes the system throughput, system-correction capacity, system-processing capacity, and system-design complexity. By using our system-performance evaluation environment, we demonstrated that the droplet-based MEFS provides higher performance, as well as lower design and integration complexity.**

***Index Terms*—Continuous-flow system, droplet-based system, microelectrofluidic systems (MEFS), polymerase chain reaction (PCR), SystemC design environment.**

I. Introduction

COMPOSITE microsystems that incorporate microelectromechanical and microelectrofluidic devices are emerging as the next generation of system-on-a-chip (SOC). Composite microsystems combine microstructures with solid-state electronics to integrate multiple coupled-energy domains, e.g., electrical, mechanical, thermal, fluidic, and optical, on an SOC. The combination of microelectronics and microstructures enables the miniaturization and integration of new classes of systems that can be used for environmental sensing, control actuation, electromagnetics, biomedical analyses, agent detection, and precision-fluid dispensing. There remain, however, several roadblocks to rapid and efficient composite system design. Primary among these is the need for modeling, simulation, and optimization tools.

The 2001 International Technology Roadmap for Semiconductors (ITRS) [1] clearly identifies the integration of electro-chemical and electro-biological microelectrofluidic systems (MEFS) as one of the five difficult challenges that will be faced beyond 2005, when feature sizes shrink below 100 nm. In fact, the ITRS document anticipates that MEFS components will be integrated in commercial SOCs beginning 2006. Therefore, there is a pressing need for innovative research in computer-aided design methodologies and design tools for MEFS to tackle the challenges that will be faced by the semiconductor industry beyond the time horizon of the 2001 ITRS document.

Microfluidic devices that combine existing components into a system are now being routinely developed. These systems are usually based on continuous-flow components, such as microvalves [2], micropumps [3] and channels [4]. There are, however, several problems inherent in such designs. These problems include complex-system architecture, dead volumes, and integration complexity. An alternative approach to the design of microfluidic systems is based on droplet actuation, which uses electrowetting-based actuation to move fluidic samples. Electrowetting-based actuation has recently been proposed for optical switching and chemical analyses [5]–[7].

DNA analysis, such as sequencing, for the detection of pathogens requires a sufficient concentration of DNA [8]. DNA samples taken from blood or tissue are often too diluted to be useful in practice. Polymerase chain reaction (PCR) is therefore used to enzymatically amplify specific DNA fragments [9]. A typical bimolecular protocol involving DNA analysis relies on the following generic steps: 1) mix sample DNA with fluorescent dyes; 2) perform PCR; 3) detect the DNA concentration using fluorimeters; 4) secondary detection and purification; and 5) DNA analysis. The focus of this paper is on steps 1–4 since they appear to be most suited for on-chip integration.

Due to the critical role that PCR plays in many biomedical/chemical applications, the design and implementation of an integrated PCR system with high throughput and low integration complexity is becoming especially important. Integrated PCR-system design is complex due to the heterogeneity of devices and coupled-energy domains. As a result, integrated PCR-system design requires a multidisciplinary approach. Behavioral modeling and simulation to evaluate the PCR-system performance is an important design step, and the processing and transportation times are basic performance parameters for this system. In this paper, we model two integrated PCR systems using SystemC and present a performance comparison for them based on simulation. The performance comparison includes the system throughput, system-correction capacity, system-processing capacity, and the system-design complexity.

Manuscript received May 30, 2002; revised September 27, 2002, May 23, 2003, and October 6, 2003. This work was supported in part by DARPA under Contract No. F30602-98-2-0140 and in part by the NSF under Grant No. EIA-0312352. A preliminary version of this paper was published in *Proceedings of the International Conference on Modeling and Simulation of Microsystems*, San Juan, PR, pp. 48–53, April 2002. This paper was recommended by Associate Editor H. A. Mantooth.

T. Zhang is with the Cadence Design Systems, Inc., Cary, NC 27511 USA.

K. Chakrabarty and R. B. Fair are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708-90291 USA (e-mail: krish@ee.duke.edu).

Digital Object Identifier 10.1109/TCAD.2004.828115

SystemC is a new open source library based on C++ [10]. It supports hardware-software codesign and the description of the architecture of complex systems consisting of both hardware and software components. It has been used in electronic hardware/software codesign [11], system-level design [12], and hardware synthesis [13]. MEFS is a new application area for SystemC. A MEFS hierarchical modeling and simulation environment based on SystemC is described in [14].

We present a performance comparison between two types of microfluidic systems—continuous-flow systems and droplet-based systems—in this paper. This comparison is carried out for the PCR using the microelectrofluidic design environment based on the SystemC programming language. We carry out behavioral modeling and simulation of PCR using this design environment. The performance comparison shows that the droplet-based microfluidic system provides better performance, yet offers the important advantage of lower design and integration complexity.

The main contributions of this paper include the following:

- a SystemC-based integrated design environment for MEFS and its application to a realistic biomedical system, namely PCR system;
- the design of a reusable continuous-flow PCR system and a droplet-based PCR system;
- a demonstration via modeling and simulation of the superior performance of the droplet-based system compared to a continuous-flow system.

The organization of this paper is as follows. First, we review the principles of DNA amplification using PCR in Section II. Section III discusses a reusable continuous-flow PCR system and its physical implementation. This reusable PCR system is based on the reconfigurable microliquid handling-system architectural design described in [15]. Section IV presents a PCR system based on droplet technology. Its physical implementation is also described. In Section V, the MEFS modeling and simulation perspectives are discussed, the fundamental variables and elements needed to describe MEFS characteristic are defined. In Section VI, a hierarchical modeling and simulation environment based on SystemC is presented. The architecture of the environment and the associated functional packages are discussed. The performance comparison between continuous-flow and droplet-based microfluidic systems is discussed in Section VII. The comparison is carried out using accurate, yet computationally-efficient, SystemC models of the PCR system. Conclusions and future work are discussed in Section VIII.

II. PCR

In this section, we present an overview of PCR, which has emerged as a powerful tool for the detection of pathogens, where the amount of sample is often small and direct detection is impossible. Its specific application is for rapid enzymatic amplification of specific DNA fragments. PCR can amplify genomic DNA exponentially using temperature cycles. Consider a DNA duplex consisting of regions $ABCDE$, as shown in Fig. 1 [8]. If the sequences of B and D are known, then millions of copies of C (the target) can be obtained by PCR. One strand of this duplex is denoted $a-b-c-d-e$, and the complementary strand is denoted with $a'-b'-c'-d'-e'$. PCR is carried out by adding the following components to a solution containing the target sequence: 1) a pair of primers, b and d'; 2) all four deoxyribonucleoside triphosphates (dNTPs); and 3) a heat-stable DNA polymerase. As shown in Fig. 1, a PCR thermal cycle consists of three steps:

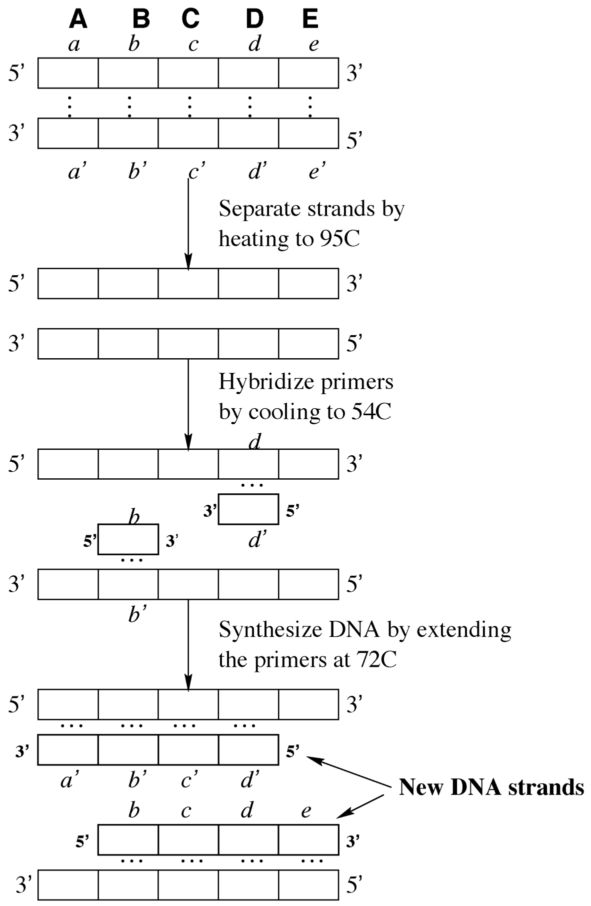

A B C D E
a b c d e
5' 3'
3' 5'
a' b' c' d' e'
Separate strands by heating to 95C
Hybridize primers by cooling to 54C
d
d'
b
b'
Synthesize DNA by extending the primers at 72C
a' b' c' d'
New DNA strands
b c d e

Fig. 1. PCR [7]. A cycle consists of three steps: strand separation; hybridization of primers; and extension of primers by DNA synthesis.

1) *Strand separation*. The two strands of the parent DNA molecule are separated by heating the solution to 95°C for 45 s.
2) *Hybridization of primers*. The solution is then abruptly cooled to 54°C to allow each primer to hybridize to a DNA strand. Primer b hybridizes to b' on one strand, and primer d' hybridizes to d on the complementary strand. This annealing process takes 30 s.
3) *DNA synthesis*. The solution is then heated to 72°C, the optimal temperature for *Taq* DNA polymerase. This allows the *Taq* DNA polymerase to attach at each priming site (where primers have annealed) and extend (synthesize) a new DNA strand. Elongation of both primers occurs in the direction of the target sequence because the $3'$ end of primer d' faces c, and the $3'$ end of primer b faces c'. One of the new DNA strands is $b-c-d-e$ and the other is $a'-b'-c'-d'$. The process takes 90 s.

In the next two sections, we describe the implementation of two integrated PCR systems: continuous-flow PCR systems and

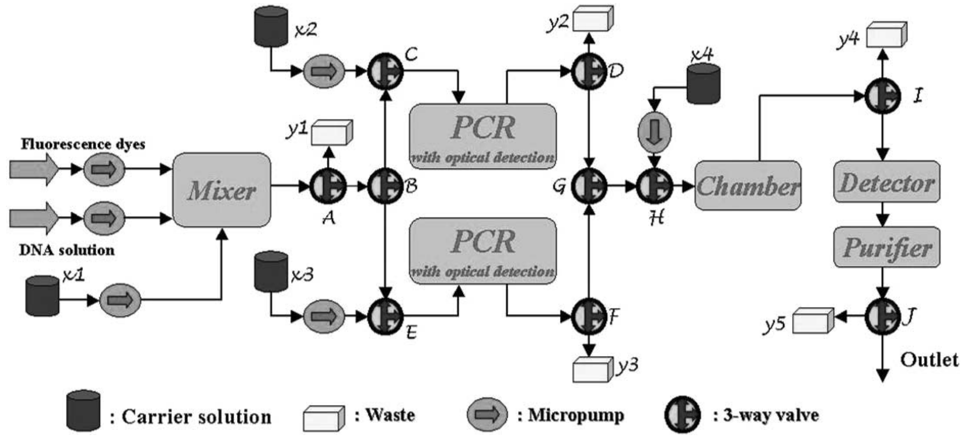


Fig. 2. Proposed reconfigurable continuous-flow PCR system consisting of three functional blocks: the preprocessing block, the processing block, and the postprocessing block.

droplet-based PCR systems. Without loss of generality, we assume that both systems have the same fluidic operational units such as detectable PCR, conductivity detector, and purifier, the differences between them lie in the mixer and the flow control subsystem. These PCR systems are then modeled using SystemC in Section VII.

III. Reconfigurable Continuous-Flow PCR System

Continuous-flow MEFS [16] use pressurized flow with mechanical flow control devices such as microvalves and micropumps [2]. As discussed in [17], the sequential process architecture suffers from poor performance. In order to address the under-utilization of system resources and improve the system yield, we propose an integrated reusable and reconfigurable PCR system. Its conceptual architecture is presented in Fig. 2. The system uses a parallel PCR configuration, and it is based on a reconfigurable microliquid handling system architectural design [15].

The reconfigurable continuous-flow PCR system consists of three functional blocks: the preprocessing block (mixers); the processing block (closed-chamber PCRs); and the postprocessing block (detectors and purifiers).

The design is based on the following considerations:

- *Independent operational units*. Due to the sequential flow between microfluidic components—the PCR, the detector, and the purifier—the system performance is limited by the slowest part of the system. Therefore, a redesigned architecture is required in which each operational unit of processing blocks can be operated independently.
- *Independent fluidic-flow cycle*. In order to make each operational unit operate independently, an independent fluidic-flow cycle has to be built for each operational unit. For instance, as shown in Fig. 2, the combination of the carrier chamber $x1$ and the waste chamber $y1$ form the mixing fluid-flow cycle. This cycle carries the incoming DNA solution and associated reagent into the mixer. In addition, the combination of $x1$ and $y2$ forms a cycle to carry the solution mixture from the mixer to a PCR. Other fluidic paths include the PCR cycle with the $x2$ carrier and the $y4$ waste chamber, another PCR path with the $x3$ carrier and the $y4$ waste chamber, and the postprocessing path with the $x4$ carrier and the $y5$ waste chambers.
- *Temporary storage buffer*. Due to the difference in processing capability (throughput) between functional blocks, and in order to enhance the system throughput, temporary storage buffers are necessary between certain function blocks. For example a storage buffer is inserted between the processing block and the postprocessing block shown in Fig. 2. The conductivity process requires a much lower flow rate than that in the transportation channel between microfluidic components. Therefore, a storage buffer is necessary to temporarily store the PCR processed product. In addition, temporary storage buffers can separate the sequentially connected processors into several independently operating functional blocks, thus benefiting system performance, and system reconfigurability. For example, the temporary storage buffer shown in Fig. 2 can separate the processing chain into the processing block and the postprocessing block.
- *Reusability*. System reconfigurable architectural design makes the system meet different performance-matrix requirements, when the system architecture is reconfigurated. One of the fundamental facets of the reconfigurable architecture is reusability. Reusability implies a number of issues with regards to cross-contamination and cleanliness. The carrier flow is necessary for each functional block. When there is an incoming fluidic solution, the carrier flow takes the solution through the process. Between the interval of two fluidic solutions, the carrier flow with the constant flow rate can function as a cleaning solution.

A. Closed-Chamber PCR Functional Block

The detectable PCR can continuously monitor the change of DNA concentration during the PCR process. The closed-chamber PCR with optical detection belongs to this category. Advantages of the PCR are its small size ($6 \times 4 \times 1.5$ mm) and the sealing of the chamber with a Pyrex wafer using the anodic-bonding method. The transparent surface of the Pyrex makes it possible to incorporate optical readout methods [18]. Because of the issue of component

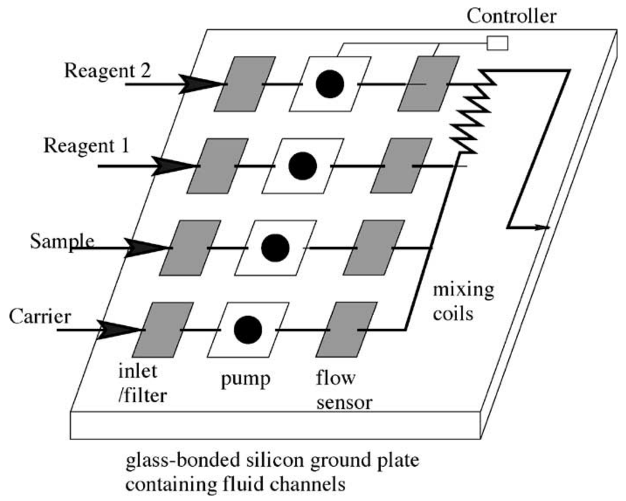


Fig. 3. PCR mixer consists of inlet units, actuation pumps, flow sensors, and mixing coils.

size, an integrated reusable continuous-flow system integrates the closed-chamber PCR on a board. On the other hand, the proposed integrated droplet-based system integrates the closed-chamber PCR on a chip for DNA thermal amplification.

After PCR is used to successfully amplify the target DNAs, the amplified DNA samples can be used for target applications, such as mutation analysis, genetic mapping and sequencing [8]. However, before subsequent analysis of the amplicon, the target DNA concentration of the PCR product has to be verified, and the PCR product has to be purified to remove unwanted salts, primers and enzymes.

Although PCR can continuously monitor the change of DNA concentration during the PCR process, this detection suffers from poor detection limits due to the limitation of UV absorbance and fluorescence [18]; hence, a backup detector is necessary. An attractive method for PCR is based on conductivity detection. The analytical response, measured by the conductivity, is related to the concentration of species [19]. As long as the analyte has a conductance different from that of the carrier solution, the DNA can be analyzed in its native state [19]. Conductivity offers several advantages compared to other common detection schemes used for DNA detection, e.g., it is amenable to small column detection. Since the detector does not have any driven power, the time for the fluidic sample passing the detector, referred to as detection time, depends on the micropump flow rate and the fluidic sample volume.

After amplification and conductivity detection, the successfully amplified DNA product needs to be purified to remove unwanted salts, primers, and enzymes for further analysis. A complete description of the PCR reverse-phase purification can be found in [9]. The volumetric flow ranges from one to several microliters per minute. In contrast to the detector, the purifier described in [9] has driven power; hence, the purification time is based on the fluidic sample characteristics.

B. Mixer

Several mixers comprise the preprocessing block in the PCR system. The basic structure of a mixer is shown in Fig. 3 [20]. The inlet unit connects the microsystem with the macroscopic environment. The micropump delivers a constant flow rate for liquids. There are two microvalves located at the input and the output of each micropump, which can prevent the reverse flow of the liquid. In addition, each micropump must be connected with a flow sensor to measure the flow rate [21].

TABLE I
DESIGN PARAMETERS AND THEIR NOMINAL VALUE FOR A MIXER

Parameters	Values
Flow rate	10μl/min
Internal pump size	7mm $\times$ 7mm [24]
Internal valve size	2 mm $\times$ 2 mm [25]
Shut-off flow rate	$\leq 0.1\mu$l/min
Mixing unit size	1mm - 4mm

In order to fully mix liquids, the average flow rate must be of the order of 10 μl/h [22]. The mixing of fluids in small channels is a major technical challenge. In a very small channel, the liquid exhibits only laminar flow. A good mixing of different liquids requires turbulence in the flow. A simple solution is to coil the route of the microchannel in the mixer. The flow rate in our system is realistically assumed to be 10 μl/min due to the coil structure and small volume of fluorescent dyes [23].

Table I shows the critical design parameters for a mixer based on a state-of-the-art design.

C. Transportation Expense

The total time for each fluidic sample staying the PCR system consists of three periods: the time of waiting for the system resources; the processing time; and the transportation time. Based on the previous discussion, the mixing time, detecting time, and the transportation time depend on the micropump flow rate.

Since an integrated continuous-flow system as shown in Fig. 2 has not been demonstrated in the literature, we base our performance analysis on the typical values of the parameters from component design. Transportation time is a very important factor influencing the continuous-flow system performance. Since the internal channel length of a three-way microvalve is 8 mm [2], the viable length of the channel between two processors is approximately equal to the number of three-way microvalves multiplied by the length of the internal channel. For instance, there are two three-way microvalves between the mixing unit and the PCR chip, thus, the channel length L between them is at least 16 mm. The regular cross-section area A of the microchannel is assumed to be 1600 μm^2 [22]. The volume of the channel between two processors is given by

$$\text{Volume} = A \times L = 1.6 \times 10^{-3} \times 16 = 0.0256\ \mu\text{l} \quad (1)$$

Although the maximum flow rate of a micropump can be 1000 μl/min, the realistic flow rate v_{liquid} is around 10 μl/min due to the limitation of the three-way microvalve and the mixer flow rate. Therefore, based on (1), the transportation time between the mixer and the PCR reaction chamber is

$$t = \frac{V}{v_{\text{liquid}}} = \frac{30\ \mu\text{l} + L \times A}{10\ \mu\text{l/min}} \simeq 3\ \text{min} \quad (2)$$

where the V is total volume of the liquid, which is equal to 30 μl, v_{liquid} is the flow rate, and L is the distance between the mixer

TABLE II
DESIGN PARAMETERS FOR THE RECONFIGURABLE CONTINUOUS-FLOW PCR SYSTEM

Design Parameters	Values
Mixing time	3 min
PCR time	4 min
Detection time	3 min
Purification time	3 min
Transportation time between pre-processing block and the processing block	3 min
Transportation time between the processing block and the post-processing block	1.5 min

and the reaction chamber. $L \times A$ is negligible when compared to the liquid volume V, hence, it can be ignored.

Table II shows the values of the design parameters for the reconfigurable continuous-flow PCR system. Based on (2), the transportation time from the mixer to the PCR is 3 min. Because only the three-way microvalves and transportation channels lie between the PCR reaction chamber and the storage buffer, the transportation flow rate can be expected to be as high as 20 μl/min without the limitation of the mixer. Therefore, the transportation time between the reaction chamber and the storage buffer is 1.5 min.

Due to the limitation of UV absorbance and fluorescence [18], the backup detector using the conductivity technique may detect that the PCR product does not match the concentration requirement. The DNA solution is then called an "unqualified solution." Because of the unidirectional fluidic flow, the processed fluid sample cannot be sent back, and the reconfigurable continuous-flow PCR system lacks system-correction capability.

IV. DROPLET-BASED PCR SYSTEM

Another physical implementation method for MEFS is based on droplet-flow technology. It is based on electrically-driven liquid handling without mechanical elements. Section IV-A introduces a droplet-based PCR, and its physical implementation is presented in Section IV-B.

A. Droplet-Based PCR System

The electrostatic actuation method has been proposed for manipulating microdroplet movement [7], [26], [27]. A potential architecture of a droplet-based PCR system is based on the electrowetting-based actuation presented in [5]. The rapid actuation of discrete liquid droplets is based on direct electrical control of their surface tension.

The droplet-based PCR system has the same architecture as the reconfigurable continuous-flow PCR system and, as we assumed, it also have the same fluidic processing blocks such as detectable PCR, conductivity detector, and purifier. The difference between them lie in the mixer and the flow-control subsystem. The DNA solution is moved into the PCR system through inlet units. The mixing units and other processor chambers are developed with electrode arrays. These arrays are used to control the fluidic flow. The transportation chain is used to connect different fluidic processing blocks. Because there are no large mechanical elements in the transportation chain, the distance between any two operational units is decreased. The direction of movement of fluidic samples is reversible.

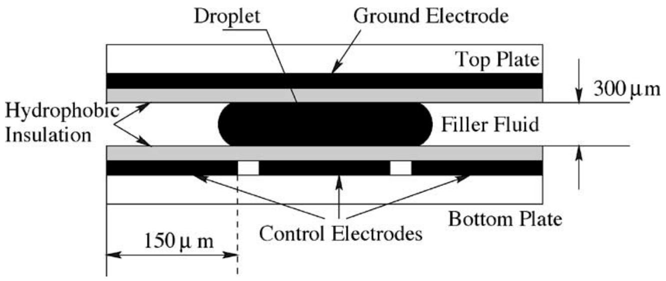


Fig. 4. Schematic cross-section of the electrowetting microactuator [5].

TABLE III
DESIGN PARAMETERS FOR THE ELECTROWETTING ACTUATOR

Design Parameters	Values
Thickness between two planes	300μm
Width of control electrodes	150μm
Maximum droplet speed	15 cm/second

B. Physical Implementation

1) Electrowetting Microactuator: The electrowetting microactuator is presented schematically in Fig. 4 [5]. A droplet of polarizable and conductive liquid is sandwiched between two sets of planar electrodes. The upper plate consists of a single continuous ground electrode, while the bottom plate consists of an array of independently addressable square-shaped control electrodes. With each electrode independently controlled, multiple fluidic droplets can be moved simultaneously. There must be at least one electrode between two droplets in order to maintain isolation.

Table III shows the critical design parameters for the electrowetting actuator [5].

2) Droplet Mixer: The concept of a "droplet mixer" [26] is that two sample droplets are fed from different inlet units, mixed with each other, and moved to the outlet. The droplet mixer concept offers the advantages of simple construction, no moving control devices, and no dead-volume. The concept of a mixer can be used for the droplet reactor design. Because of the agitation during the droplet movement, the mixing time can be largely reduced. Normally, the time for one μl droplet mixing is several seconds. Because electrodes can be controlled independently, a large-size droplet can be separated into several smaller droplets, and mixed simultaneously, the mixing time of a droplet is not related only to the volume. It is reasonable to assume that the mixing time for a 30 μl sample is 10 s [5].

3) Transportation Expense for a Droplet-Based PCR System: The topology of the grid array influences the route that a droplet must take in moving from one port to another. Hence, the definition of the distance between two ports must consider the structure of the droplet-based PCR system. The distance between two ports is defined as the number of the electrodes between them, n. The number of droplets, n_{droplet}, whose volume V_{droplet} is 1 μl, existing in the 30 μl liquid sample is

$$n_{\text{droplet}} = \frac{V_{\text{liquid}}}{V_{\text{droplet}}} = \frac{30\,\mu\text{l}}{1\,\mu\text{l}} = 30 \qquad (3)$$

where, V_{liquid} is the volume of a sample. Since the droplets can be moved simultaneously at a maximum speed of v_{droplet}

TABLE IV
DROPLET-BASED PCR SYSTEM-DESIGN PARAMETERS

Design Parameters	Values
Mixing time	10 seconds
PCR time	4 min
Detection time	3 min
Purification time	3 min
Transportation time between any two functional blocks	8 seconds

15 cm/s (which is equal to 8.2 electrodes/s, the width of the electrode is 1.82 mm), and the moving droplets must have at least one spare electrode between them in order to keep isolation; therefore, the equation to calculate the transportation time t of the liquid through n electrodes can be obtained as

$$t = \frac{\left(n + 3 + 2 \times (n_{\text{droplet}} - 2)\right)}{v_{\text{droplet}}}. \tag{4}$$

The inductive steps to derive (4) are as follows.

Without loss of the generality, the value of v_{droplet} is assumed to be 1 electrode/s, and the number of electrodes n is assumed to be 4.

1) We first show that (4) holds for $n_{\text{droplet}} = 1$ and $n_{\text{droplet}} = 2$.

   It is obvious that the total transportation time for one droplet passing through 4 electrodes is 5 s.

   From (4), we obtain $t = (n + 3 + 2 \times (n_{\text{droplet}} - 2))/v_{\text{droplet}} = (4 + 3 + 2 \times (1 - 2))/1 = 5$ s.

   Since the moving droplets must have at least one spare electrode between them, in order to maintain isolation, when the second droplet arrives at the fifth electrode, the first droplet just arrives at the seventh electrode. Therefore, the transportation time is 7 s.

   From (4), we obtain $t = (n + 3 + 2 \times (n_{\text{droplet}} - 2))/v_{\text{droplet}} = (4 + 3 + 2 \times (2 - 2))/1 = 7$ s.

2) Suppose that (4) holds for $n_{\text{droplet}} = m$. no. $t = (n+3+2 \times (n_{\text{droplet}} - 2))/v_{\text{droplet}} = (n+3+2 \times (m-2))/1 = n + 2 \times m - 1$.

3) We next show that (4) holds for $n_{\text{droplet}} = m + 1$.

   Based on the above analysis, it easily follows that when the number of droplets increases by 1, the transportation time is increased by 2, and the transportation time becomes $n + 2 \times m + 1$.

From (4), we obtain $t = (n+3+2 \times (n_{\text{droplet}} - 2))/v_{\text{droplet}} = (n + 3 + 2 \times (m + 1 - 2))/1 = n + 2 \times m + 1$. This proves the result.

Since the distance between any two ports is within several millimeters, the number of electrodes, whose width is 1.82 mm, between two ports can be assumed as 10. The transportation time of a 30 μl sample between any two ports is approximately as follows:

$$\begin{aligned} t &= \frac{\left(n + 3 + 2 \times (n_{\text{droplet}} - 2)\right)}{v_{\text{droplet}}} \\ &\simeq \frac{\left(10 + 3 + 2 \times (30 - 2)\right)}{8.2} \\ &\simeq 8 \text{ s.} \end{aligned}$$

Table IV shows the values of the design parameters for the droplet-based PCR system. The droplet-based PCR system offers high transportation speed and inherent parallelism; hence, we ignore the time for splitting a droplet. The mixing time tends to dominate splitting and transportation times.

V. MEFS Hierarchical Modeling and Simulation Perspective

MEFS complexity arises due to the growing number of devices and the increasing levels of heterogeneous coupled-energy domains. As a result, system design requires a comprehensive model to study the dynamic behavior of the system. The modeling of MEFS behavior consists of two integral parts: system-level modeling and component modeling.

System-level modeling involves performance modeling and behavioral simulation for specific biomedical and chemical applications. In contrast, component modeling investigates the individual microfluidic component behavior, and emphasizes the definition of physical properties and relationships at the component level. Component modeling offers an approach that is complementary to system-level simulation. A MEFS closed-loop integration design environment should extend system design from the component level to the system level.

In this section, the fundamental variables and elements needed to describe MEFS characteristics are defined from the lower component level to the higher system level. These fundamental variables capture the MEFS behavior, and they are critical requirements for an appropriate MEFS modeling and simulation language.

A. MEFS Dynamic Modeling and Simulation at Component Level

1) Classification of Dynamic System Models: Mathematical models are needed to study the dynamic behavior microsystems. These models can be classified into two categories based on the nature of the underlying differential equations [28]: 1) Distributed-element models and 2) Lumped-element models.

A wavelength/physical size concept can be used to explain the rationale of building lumped-element models for any physical system that exhibits wave propagation, such as electromagnetic systems, mechanical vibrating systems, and acoustic systems. The key concept here is that if the physical size of a device is small compared to the wavelength associated with signal propagation, the device may be considered lumped, and a network lumped-element model can be employed. Typically, microelectrical systems can be treated with the simple lumped-element approach. In addition, the essential solid body characteristics allow micromechanical systems to be analyzed as lumped-element models [29].

However, in contrast to the electrical and mechanical energy domains, matter and energy may not be continuously distributed over space within some fluidic systems. In addition, due to the generally less well-defined shapes of bodies of fluid (as compared to solid bodies), microfluidic systems appear to be less suited for the lumped-element viewpoint. Since every spatial location has its own flow rate and direction of flow, using a given spatial point to be representative of the local environment may cause behavioral description errors. Nevertheless, using a lumped-element model to describe microfluidic systems is appropriate when the fluidic flow is laminar, the fluid is incompressible, and the fluid shape is well defined [28]. For instance, when a fluidic sample flows in a channel whose diameter is very small (millimeters), the fluidic flow can be described using the lumped-element model. Within a given element, there is no variation, but behavior such as pressure and velocity usually change between different elements. It is clear that when a model is made up of a number of smaller elements, the stepwise variation nearly approximates the true smooth variation. Based on several researchers' experiences [25], [28], [30], it is now recognized that when studying fluidic movement in a microfluidic system, lumped-element models can provide good results if the fluidic element is concentrated into ten elements per wavelength at the highest operating frequency.

Therefore, after considering the fundamental characteristics and their relationships to multiple energy domains, we conclude that lumped-element models with ODAEs are appropriate for describing and studying dynamic MEFS behavior involving multiple, coupled energy domains.

2) Fundamental Variables: The fundamental variables are required to describe the system dynamic behavior. The physical quantities in multiple energy domains can be viewed as types of single-port element variables: *across* and *through*. These two variables are used to describe the power and energy-flow variables, respectively. An *across* variable denotes a difference in a physical condition across the terminals of an element. A *through* variable denotes a physical quantity transmitted through the terminals of an element. The power flow through a port into a fluid system can be expressed with the through variable and the across variable: volume fluid-flow rate Q and fluid-pressure drop P. The volume $V(t)$ represents the total volume of fluid passing through the port over a given time period. The pressure momentum $\Gamma(t)$ is the time integral of pressure, which is analogous to the momentum in mechanical systems [28]. A modeling and simulation language needs to have the capability to describe these fundamental variables and their constitutive relations to study the system behavior.

3) ODAEs Solver: Due to the complexity of MEFS designs, it is better to relieve the system designer from the burden of simulator development. Designers should mainly focus on the system modeling using related modeling and simulation languages. The associated simulator can automatically solve the system model with sophisticated mathematical methods, and it can offer a flexible and standard interface for a user-defined program.

B. MEFS System-Level Modeling and Simulation

1) MEFS Behavior Modeling Perspectives: The MEFS system-level modeling requires a single language to describe different behavioral perspectives: 1) discrete event-scheduling perspective; 2) discrete process-interaction perspective; and 3) continuous perspective. Multiple perspectives can be combined, allowing portions of the dynamical behavior to be described by a discrete modeling paradigm and other portions of the dynamical behavior to be described by a continuous modeling paradigm.

Although the basic MEMS dynamic behavior is described using event-scheduling, continuous or both perspectives, this modeling paradigm cannot directly represent MEFS behavior. MEFS behavior not only requires a combination of discrete and continuous perspectives, but it also requires the discrete representation including event-scheduling, process-interaction, and a combination of both. For instance, the event-scheduling perspective is necessary to model a microfluidic sample arrival event. The continuous perspective is used to describe a thermal reaction involving solution mixtures, and the energy-conservative queuing nature of a biochemical DNA-analysis system must be described with a process-interaction perspective [31].

2) Object-Oriented and Dynamic Data Structure: In contrast to the MEMS processor-oriented modeling perspective, the MEFS system-level design mainly focuses on the change of fluidic sample characteristics. Therefore, the process-interaction perspective is popularly adopted to describe this fluidic sample-oriented MEFS behavior. In addition, in order to more effectively represent the features of each fluidic sample in MEFS, a complex but flexible data structure is necessary. The features are defined in the following.

- *Fluidic sample property*. This item presents fluidic sample physical and chemical features, such as the fluidic sample volume and the sample temperature, etc. These features may be changed during the fluidic sample processing period.
- *System resource utilization*. This item records the status of each fluidic sample using system resources. For instance, which processor is used by that fluidic sample? Which channel is used to deliver that fluidic sample from the storage buffer to the processor or from the processor to outlet?
- *Fluidic sample simulation clock*. This item records the simulated time value of each process event for a certain fluidic sample. For example, the time value when that fluidic sample arrives at the input to the handling system, the time value when that fluidic sample arrives at a storage buffer, the thermal reaction time for that fluidic sample, etc.

By definition, stochastic systems exhibit runtime nondeterminism; any particular simulation is simply one observation of the random behavior. Thus, data structures associated with random variables cannot generally be predefined at model development or instantiation, and dynamic data structures are required that can be modified during runtime of the model, i.e., simulation. A dynamic data structure, such as a linked-list, can be an effective data representation for a set of objects or data values, where set membership can vary during simulation by creating and destroying objects. For instance, a linked-list is often used to represent a set of fluidic samples waiting for service or a set of jobs waiting for execution [32]. In addition, this dynamic data structure can describe more complicated queuing behavior, involving priorities, preemptions, redistributions, and terminations [33].

3) User-Definable Behavior: One of the most effective ways to address the difficulties of MEFS design complexity is to create abstractions at the system level. These abstractions highlight relevant system characteristics and deemphasize or hide all other information. They also reveal how a designer views the intent and operation of a complex system. Thus, system performance modeling languages are required to provide a basic set of predefined functions and behaviors to construct application-specific, user-definable abstractions [34]. In addition, because of the complexity of MEFS architecture, the different functional blocks can be connected to each other sequentially or in parallel, either the sequential process contains the concurrent procedure, or the concurrent procedure embodies the sequential process.

This aspect of system-level modeling can be realized using a variety of constructs and statements supporting both sequential and concurrent executional semantics for procedural and parallel tasking and methodologies, respectively. In this manner, a system performance modeling language is molded to fit an application rather than the counter situation of contorting an application to fit an inflexible system performance modeling language.

4) Time-Advanced Mechanisms: Due to the stochastic nature of MEFS application behavior, a variable is necessary to keep track of the current value of simulated time when the simulation proceeds. This variable is called the *simulation clock*. It is also useful to advance simulated time from one value to another for the event-scheduling perspective. In addition, it is necessary to synchronize events in the simulation. Clocks order events in time so that parallel events are properly modeled by simulator on a sequential computer. There is generally no relationship between simulated time and the time needed to run a simulation on a computer.

5) Scalability of Simulation: Because of the hierarchical structure of MEFS, MEFS simulation requires the study of design scalability. Existing MEMS hierarchical modeling and simulation techniques focus on low-level components. The entire MEMS component as a single behavior entity forms the top level of hierarchy, and the constituent MEMS elements, such as plate masses and beam springs, form the hierarchical lower level [35]. However, this down-top approach is not efficient for MEFS. A scalable methodology with top-down decomposition and down-top verification is necessary for MEFS, it must handle heterogeneous, multiple-component systems, and address complex fluidic-application and mixed-level component simulation. It is important to investigate how the performance of a microliquid handling system architecture scales with increasingly complex chemical and biological analyses, and what types of biomedical applications can be practically miniaturized via microfluidic molecular processing. In addition, it is also necessary to investigate how the performance of the microliquid handling system scales with advances in constituent microfluidic device technology. Therefore, the MEFS system-level modeling and simulation languages must possess a hierarchical scalable-design capacity.

6) Statistical Analysis Capacity: The purpose of simulation is to imitate the operation of a real-world system, and then to use the resulting simulation output data to infer the real-world system functionality and performance. MEFS high-level system performance models are generally stochastic because either the system is too complex to be analytically characterized, design details are unknown, or overall performance depends on ambient factors that are nondeterministic [36]. Stochastic systems dynamically vary over time because the system operation is dependent on one or more random variables. Hence, the resulting simulation output data exhibit random variability. Consequently, the statistical analysis approach is very important [33]. Statistical analyses require the language capacity to compile various usage information during system execution to estimate the mean, variation, correlations, and confidence intervals of the sampled random results. Probabilistic and statistical analyses also require multiple data types, powerful mathematical resources (function libraries), and operating system storage (file) input/output.

Overall, the lumped-element models with ODAEs are appropriate to describe the MEFS component-level dynamic behavior coupled with multiple energy domains. The distributed-element models with PDAEs are also necessary to study the lumped-element models' possibility, accuracy, and limitation. These component-level modeling and simulation requires the simulation languages to have the capacity to describe the MEFS lumped-element and possible distributed-element models, and to represent its *across* and *through* variables and their general constitutive relations. In addition, the MEFS system-level hierarchical modeling and performance evaluation require the description capacity of simulation languages for MEFS system-level hierarchical modeling, simulation, and statistical analyzes.

VI. Modeling and Simulation Languages

Traditionally, several modeling languages and simulators have been used to support various phases of system specification, architectural design, and functional unit design. Performance-modeling languages, such as SIMSCRIPT II.5, SLAM II, and general purpose software programming languages such as C and Ada, are used for the high-level architectural design, stochastic performance analysis, and biomedical/chemical process flow simulation [33]. On the other hand, logic-modeling languages, such as VHDL/VHDL-AMS, are used for low-level functional unit design [37]. However, this system-design approach requires human intervention. It also leads to problems of misinterpretation of concept specifications in the translation between different data models and tools. Thus, it is beneficial to construct a hierarchical system modeling and simulation environment using a common system-description language and associated simulation engine, rather than multiple languages and simulators that span different levels of abstraction. The potential benefits of this approach include reductions in design time and life-cycle maintenance costs.

Thus, it is necessary to construct a hierarchical system modeling and simulation environment using a common system description language and associated simulation engine, rather than multiple languages and simulators that span the different abstraction level. In this section, a hierarchical modeling and simulation environment based on SystemC is presented. While the main focus of this paper is on the comparison between continuous-flow and droplet-based PCR systems, SystemC is pre-

TABLE V
COMPARISON BETWEEN THE FEATURES OF DIFFERENT SIMULATION LANGUAGES

Languages	SystemC	VHDL and VHDL-AMS	SLAM	Matlab	C/C++
Associated Simulator	N/A	Good	Average	Good	N/A
Component-level Description Capability	Good	Average	Poor	Poor	Good
Behavior Modeling Perspective	Good	Average	Good	Poor	Good
Data Structure	Good	Average	Average	Average	Good
User-Definable Behavior	Good	Good	Average	Poor	Poor
Concurrency/Timing Mechanism	Good	Good	Good	Poor	Poor
Multi-level Description Capacity	Good	Poor	Poor	Poor	Good
Analysis Capability	Average	Average	Good	Good	Average

sented here as an appropriate candidate to facilitate this comparison via simulation and performance evaluation. Based on the discussion in Section V, we examine the suitability of several simulation languages for MEFS hierarchical design. These languages include VHDL-AMS, SLAM, C/C++, Matlab, and SystemC. Next, SystemC is proposed as a potential candidate for complete system modeling and simulation. In addition, a hierarchical modeling and simulation environment based on SystemC is presented. The architecture of the environment and the associated functional packages are discussed.

A. Overview of Languages for Hierarchical Design

1) VHDL-AMS: Recently, VHDL has been extended to enable descriptions of continuous-time systems. The combination of discrete and continuous time language constructs are collectively referred to as VHDL-AMS [37]. VHDL-AMS supports component-level modeling and simulation of continuous and discrete systems with conservative and nonconservative semantics of energy. The equations describing the conservative aspects of a system do not need to be explicitly annotated by the user. The VHDL-AMS solver automatically verifies the conservation of energy. Although it has recently been used for MEMS design [38], it appears that the processor-oriented modeling perspective of VHDL-AMS limits its applicability for MEFS fluidic-sample oriented analysis. For example, it is difficult to use VHDL-AMS to develop powerful yet flexible data structures to describe the fluidic sample characteristics discussed in Section V. In addition, VHDL-AMS, which supports ODAEs, may not be suitable to directly describe PDAE's needed for the MEFS distributed-element models. Moreover, VHDL-AMS is not normally used to describe the system-level model behavior because of its component-level-oriented modeling perspective.

2) Performance Language—SLAM: SLAM is a high-level performance modeling language [33]. It provides the capacity to describe the overall system as a stochastic system. An important aspect of SLAM is that alternate modeling methodologies can be combined within a single simulation model. In addition, it provides several statistical reports for final data analysis, and it also provides a useful simulation methodology for performance evaluation. However, it lacks the capacity to model and simulate hierarchical multiple-level MEFS behavior. Its modeling capability is limited to abstract high-level models, and it does not support component-level coupled-energy descriptions.

3) C/C++: C/C++ are popular, powerful and flexible languages, and a wide variety of C/C++ compilers and helpful accessories are available. They provide powerful dynamic data structures. In addition, flexible semantics and adequate mathematic functions make it possible to build a wide variety of system models. However, standard C/C++ does not possess the description capacity to directly study MEFS component-level coupled-energy behavior. For example, there is no natural way in C/C++ to represent constrained data types, concurrency, and clocks. In addition, the C/C++ language does not provide an associated simulator, the designer is required to build the model solver.

4) Matlab: MATLAB is a powerful high-level language that is especially suitable for demonstrating mathematical concepts. Matlab offers a useful working environment for quick model calculation and full simulation tasks. However, based on the requirements for the hierarchical modeling and performance evaluation of MEFS, from the higher biomedical level to lower component level, Matlab lacks the capacity to model and evaluate the hierarchical performance of the MEFS architecture. In addition, Matlab does not directly support component-level coupled-energy domain modeling. It lacks the capacity for discrete event-driven modeling and concurrent simulation, and there is general consensus among most Matlab users that certain Matlab programs run extremely slowly.

5) SystemC: SystemC is a new open source library in C++. SystemC and standard C++ development tools can be used to create a system model from the system level to the component level, quickly simulate to validate and optimize the design, explore various algorithms, and provide the hardware and software development team with an executable specification of the system.

SystemC provides *Module* and *Process* to describe the complex MEFS hierarchical architecture. In addition, SystemC supports a rich set of port and data types. They are very useful to describe the different fluidic sample properties and communication between different fluidic components. The multiple-level abstract design methodology is one of the most important properties of SystemC, ranging from the higher system level to lower component level. Moreover, to model and simulate continuous perspective with SystemC, differential equations with respect to time can be discretized and transformed into corresponding difference equations.

A drawback of SystemC is that it does not provide an associated simulator, the designer is required not only to model the system behavior, but also to build the model solver. However, SystemC's procedures allow us to describe ODAEs and PDAEs easily, and then solve these equations.

In summary, as qualitatively shown in Table V, while evaluating the suitability of these languages for MEFS hierarchical design, we found that SLAM II, C/C++, and Matlab are not suitable to handle the problem of MEFS modeling and simulation. VHDL-AMS is a potential candidate; however, based on the authors's experience, it is not easy to use VHDL-AMS to model and simulate the MEFS system. Therefore, we recommend that SystemC is a viable candidate to develop a MEFS hierarchical modeling and simulation environment, even though it is the user's responsibility to build the associated simulator [14].

B. System-Level Modeling Package

System-level modeling involves the system-performance modeling and the simulation of stochastic behavior inherent in the execution of a specific biomedical and chemical application. In addition, system-level modeling studies the reconfigurable system-architecture performance, scheduling, and throughput.

1) MEFS Behavioral Description: Depending on the system-level characteristics of MEFS, the fundamental elements for system modeling include: 1) the storage part, which is used to temporarily store the fluidic samples, and examples of which includes fluidic input buffers and containment reservoirs; 2) the transportation part, which is used to deliver fluidic samples from one site to another; and 3) the processor part, consisting of fluidic analyzers and mixers, which are key for a MEFS bio/chemical application. All these functional blocks are defined using *processes*. In addition, the basic elements include: 4) the timing clock to synchronize simulation events and 5) a complex but flexible fluidic sample data structure. It contains the fluidic sample's physical properties and simulation procedure records.

2) MEFS Architectural Description: The *Master* and *Slave* processes, which can perform data transactions based on an address, are used to define the fluidic transaction between different functional blocks. *Module* and *Process* can be used to reflect the low-level parallelism of microfluidic components and the high-level ordering of procedures and functions. By combining these two mechanisms, system designers have the flexibility to model the system behavior, instead of redesigning an application to fit an inflexible system performance modeling language.

Furthermore, a mathematical package is built for MEFS system-level stochastic behavior. It contains the common real constants, and common real-probability functions. SystemC supports the capacity to build this mathematical package with the regular function procedures.

C. Component-Level Modeling Package

The goal of MEFS component modeling and simulation is to study individual microfluidic components at the component-level of abstraction, emphasizing the definition of physical properties and their relationships across multiple energy domains.

1) Energy-Domain Behavior Declarations: Based on the microfluidic component modeling common issues [14], coupled energy component modeling, which is Kirchoffian in nature, requires the declaration of specific variables to represent individual energy domains and disciplines. Similar to the energy declaration in VHDL-AMS, SystemC makes declarations for the variables for each energy domain. These declarations use the *signal* construct of SystemC.

2) Coupled-Energy Modeling and Simulation: The coupled-energy problems in MEFS, which require simultaneous statements describing concurrent events, can be addressed using the *Process* construct. We make use of three different types of Process—*Methods*, *Threads*, and *Clocked Threads*. Since the concurrent processes in SystemC are loosely coupled, the sensitivity list for each process has to be expressed explicitly. Moreover, in contrast to VHDL-AMS, SystemC does not directly provide constructs for defining energy-conservative sets of simultaneous ODAEs. It is the user's responsibility to write and verify the energy-conservative models. SystemC does not directly provide an associated simulator to solve simultaneous ODAEs over a series of intervals denoting a period of time. Nevertheless, by using the regular function procedures or *Process*, users can code various DAEs solvers with SystemC, such as derivative and integral, and add them into a SystemC component behavior model. Moreover, besides the original simulation clock, SystemC can supply a higher-frequency clock to provide a series of time intervals for more accurate ODAEs function solutions. Here, we use the relaxation-based numerical integration techniques [39] coded in SystemC to solve these ODAEs. An example of solving the ODAEs which represent the microvalve behavior is given in the next section. Note that it is difficult to implement an efficient solver for all biomicrofluidic applications.

Fig. 5 shows the program structure of a general MEF system. Each function block is hierarchically connected to the higher-level program. The connection between different functional blocks is defined on the higher level. Associated numerical simulation package and optimization package support the system modeling, simulation, and optimization [40], [41].

VII. Comparison Between Continuous-Flow PCR and Droplet-Based PCR

In this section, we compare two types of PCR systems: continuous-flow PCR systems and droplet-based PCR systems. The evaluation is based on the system-design complexity, system throughput, system-resource utilization, and system-correction

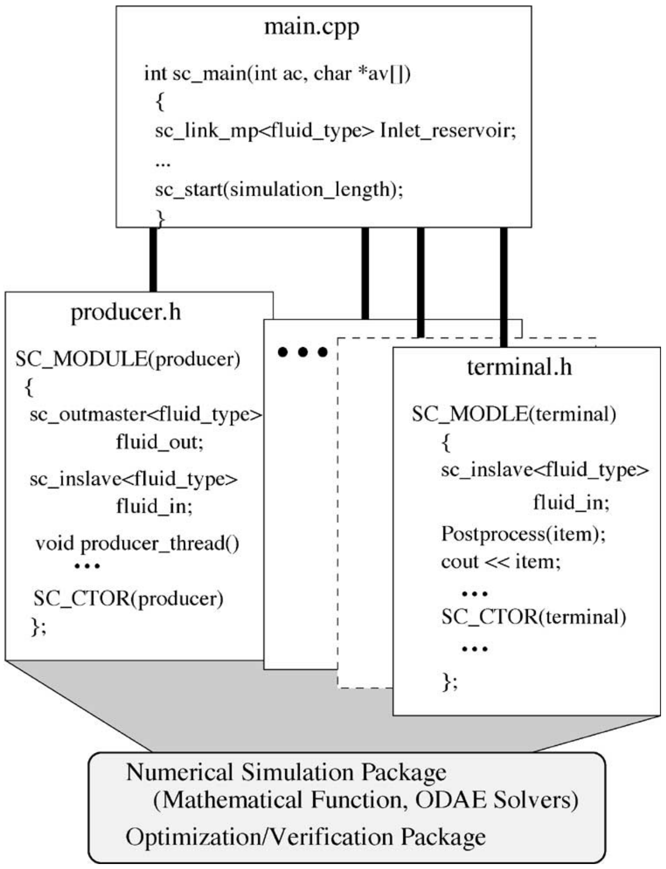

main.cpp
int sc_main(int ac, char *av[])
{
sc_link_mp<fluid_type> Inlet_reservoir;
...
sc_start(simulation_length);
}
producer.h
SC_MODULE(producer)
{
sc_outmaster<fluid_type> fluid_out;
sc_inslave<fluid_type> fluid_in;
void producer_thread()
•••
SC_CTOR(producer)
};
•••
terminal.h
SC_MODLE(terminal)
{
sc_inslave<fluid_type> fluid_in;
Postprocess(item);
cout << item;
•••
SC_CTOR(terminal)
•••
};
Numerical Simulation Package
(Mathematical Function, ODAE Solvers)
Optimization/Verification Package

Fig. 5. Description of a general MEF system model based on SystemC.

capacity. PCR system modeling and simulation are based on the SystemC design environment [14].

A. System Modeling

As shown in Fig. 5, SystemC can be used to hierarchically model general MEF systems. These models can target the biomedical process-flow level, the complex reconfigurable system-architecture level, and the more detailed and lower component level. We next present the general PCR system architecture model and an example of the function-block model. More details on SystemC modeling for MEFS are presented in [17].

1) System Architecture Modeling: The architectural simulation model of the PCR system is developed using a combination of process, event, and continuous control paradigms. The operation of the PCR system is modeled by the flow of entities (clients) through a network structure consisting of nodes and branches denoting resources, queues denoting resources, activities, and entity flow decisions.

Without loss of generality, Table VI shows a top-level program structure of the PCR system based on SystemC. This program structure also shows the general queuing network property of the PCR system. The complete PCR process can be separated into five stages based on the process routine. The first stage, the initial sample creation, is coded in *producer.h*. The second stage is the mixing stage. In this stage, the fluidic samples are moved from containment reservoirs into the mixer that is not full. The behavior of this stage is coded in *mixing.h*. When the PCR-processor block is free, samples are transported to the appropriate processor. The procedure, *processor.h*, is used for this stage. It consists of all PCR processors. After processing, the fluidic samples go to the fourth stage, the postprocessing block: detection and purification. The procedure, *post-process.h*, is used for this stage. At the end, the targeted liquid sample is transported to the terminal stage. This terminal stage is coded in *terminal.h*. Simulation results also are recorded in this stage for data analyzes.

2) Functional Block Modeling: The PCR function units consist of three processing steps:

- backward communication between PCR processors and previous function blocks, such as mixers and channels;
- forward communication between PCR processors and the next processing blocks, such as detectors and purifiers;
- PCR thermal cycle amplification processing.

The forward and backward communication processes require the definition of the handshaking protocol between different function blocks. The $sc\_link\_mp\ \langle fluid\_type\rangle$ in SystemC can be used to define communication signals. The related definition of the synchronization clock is also needed. Another smaller interval clock is necessary for the calculation related to the PCR thermal cycle process. Fig. 6 shows these signal definitions.

Fig. 7 shows the header and implementation code for the PCR functional block. The behavior of the thermal cycle process can be defined using the general mathematical function, or complex ODAEs depending on the design of the PCR system.

3) Microvalve Lumped-Element Nodal Modeling: MEFS-performance analysis is difficult because coupled-energy behavior creates strong links between high-level architecture and low-level component design parameters. We adopt the strategy of trading-off behavioral fidelity with the efficiency of analysis, "blinding" unnecessary low-level detail, and paying more attention to certain tractable subsystems [42]. All operational units need the nodal modeling to study their detailed behavior; here, we focus only on the microvalve nodal model to study the flowrate.

The pressure-driven check valves significantly affect the behavior of the micropump since they determine the micropump flow rate. The major parts of the check valve are a cantilever beam and valve seats. Normally, the cantilever lies against the valve seat, thereby closing the port to fluid flow. During operation, the fluid flow exerts the pressure against the cantilever. The cantilever, acting like a spring, deflects and allows the fluid to flow through the valve. The flow rate Φ can be treated as a function of pressure difference p and the displacement y that is the distance between the cantilever and the valve seat [25]

$$p = \sum_{i=I}^{V} \Delta p_i(\Phi, y). \tag{5}$$

The displacement y is determined partly by the pressure difference between the valves p. The behavior of the cantilever can be described by the following second-order differential equation

$$m\ddot{y} + d\dot{y} + ky = pA = \sum_{i=I}^{V} \Delta p_i(\Phi, y)A \tag{6}$$

TABLE VI
TOP-LEVEL STRUCTURE OF A GENERAL PCR SYSTEM BASED ON SYSTEMC. IT DEFINES THE COMMUNICATION PROTOCOL BETWEEN THE FUNCTIONAL BLOCKS

```
#include "systemc.h"

#include "fluid.h"
-- define fluidic features.

#include "config.h"
-- define system architecture.

#include "producer.h"
-- generate fluidic samples.

#include "ppu.h"
-- define mixer block.

#include "channel.h"
-- define channels.

#include "processor.h"
-- define process elements.

#include "post-process.h"
-- define post-process block.

#include "terminal.h"
-- define terminal nodes.
```

```
int sc_main(int ac, char *av[])
 {
 //Signals
 sc_link_mp<fluid_type>
              producer_mixer;
 sc_link_mp<fluid_type>
              mixer_PCR;
    ....
 sc_clock genClk("genClk",
             1, 0.5, 0.1, true);
    ....
 // instantiate functional blocks.
 // and connection channels.
 producer fluid_producer("Master");
    ....
 reservoir reservoir("Reservoir");
    ....
 mixing_unit mixers("MixingUnit");
    ....
 processor processor_group("ProcessorBlock");
    ....
 post-process postprocess("PostBlock")
    ....
 terminal Terminal("TerminalNode");
    ....
 sc_start(simulation_length);
 return 0;
 }
```

```
#include "systemc.h"

// connection between mixer and PCR.
sc_link_mp<fluid_type> mixer_PCR;
...

// connection between PCR and purifier.
sc_link_mp<fluid_type> PCR_purifier;
...

// Clock used for general simulation
sc_clock genClk ("genClk", 1, 0.5, 0.1, true);

// Clock used for math calculation
sc_clock mathClk ("mathClk", 0.1, 0.5, 0.1, true);
...
```

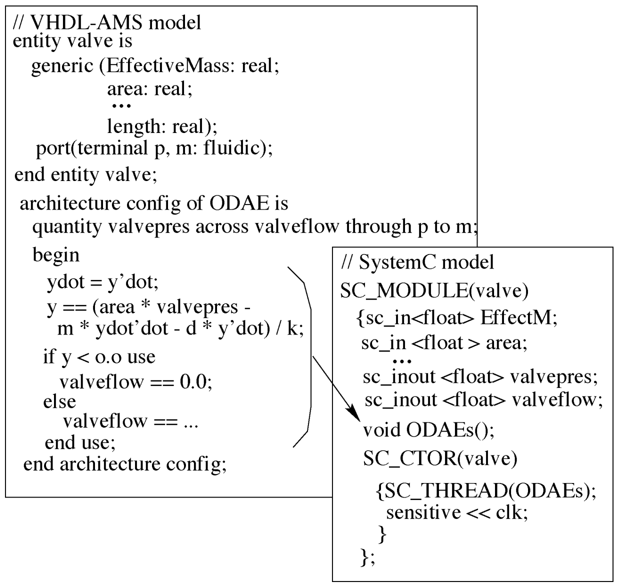
Fig. 6. Definition of the communication protocol between the PCR and related function blocks.

```
SC_MODEL( PCR )
 {
  //ports definition
  sc_out<int> empty;
  sc_inslave<fluid_type>  fluid_read;
  sc_outmaster<fluid_type> fluid_write;
  sc_in_clk wclk;

  // slave methods
  void Read_Liquid()
    {
     item_write = fluid_read;
    ... }
   void Write_Liquid()
    {
     fluid_write = item_write;
    ... }
   void Thermal_cycle()
   {...}
 };
```

Fig. 7. PCR functional block model based on SystemC.

where m is the effective mass of the cantilever, including the mass of cantilever and that of the liquid surrounding the cantilever. The parameter d is the damping constant, determined by the geometry of the cantilever. k is the spring constant described by the geometry of the cantilever, and product materials. We build this microvalve analytical model with SystemC. Since the cantilever model is inherently nonlinear and coupled, we solve it numerically. Fig. 8 shows the microvalve model coded by VHDL-AMS and SystemC, respectively. Since SystemC does not provide an associated simulator, the simultaneous ODAEs are combined with ODAE solver and solved by a *Process*: ODAEs(). Fig. 9 illustrates the general signal communication between the process and the ODAE solver. *start_trigger* signal triggers the ODAE solver so it starts to solve the ODAE



```
// VHDL-AMS model
entity valve is
   generic (EffectiveMass: real;
            area: real;
            ...
            length: real);
   port(terminal p, m: fluidic);
end entity valve;
 architecture config of ODAE is
   quantity valvepres across valveflow through p to m;
   begin
     ydot = y'dot;
     y == (area * valvepres -
       m * ydot'dot - d * y'dot) / k;
    if y < o.o use
      valveflow == 0.0;
    else
      valveflow == ...
    end use;
 end architecture config;
```

```
// SystemC model
SC_MODULE(valve)
  {sc_in<float> EffectM;
   sc_in <float > area;
      ...
   sc_inout <float> valvepres;
   sc_inout <float> valveflow;
   void ODAEs();
   SC_CTOR(valve)
     {SC_THREAD(ODAEs);
       sensitive << clk;
     }
   };
```

Fig. 8. Microvalve model coded by VHDL-AMS and SystemC.

equations, the higher-frequency clock is provided for the ODAE solver calculation. After the ODAE solver reaches the final solution, *complete_signal* changes to 1, and the process continues.

Table VII shows the microvalve-determined design parameters and their design value. In addition, the fluid density and viscosity for each fluidic sample are assumed to be the same. The microvalve operating frequency is set at 100 Hz. Since the turn-on and shut-off times of the cantilever of the microvalve is trivial compared to the transportation time, the delay of the microvalve is ignored.

4) Stochastic Acquisition Assumption: The PCR system can process a series of DNA solutions. The DNA solutions are sequentially moved into the system, amplified, detected, and purified, At the end, the processed DNA solutions are moved out for future processing, such as mutation analysis and genetic mapping [8]. The biomedical and chemical application always possess the stochastic behavior, especially the fluidic sample acquisition [31].

Without loss of generality, the volume for each DNA solution is assumed to be the same and equal to 30 μl. In addition, the acquisition function, $f(x)$, for the DNA solution to the PCR system is modeled by a traffic of liquid samples separated by interarrival times, denoted by $\{t_1, t_2, \ldots\}$. These interarrival variables are independent, identically distributed (IID) random variables, and they are characterized by an exponential probabilistic distribution given by (7), with a mean value of 4 min. That is $\lambda = 1/240$ (Note the basic system time unit is a second). The incoming DNA solution is moved into the system until the associated system resources are available. The general C language-based mathematical package can provide the common real constants, and common real probability functions. SystemC supports the capacity to build this mathematical package with the regular function procedures. More details can be found in [31]

$$f(x) = \lambda e^{-\lambda x} \quad x \geq 0. \tag{7}$$

B. System-Design Complexity

A reconfigurable continuous-flow PCR system needs a total of 16 mechanical flow control devices: ten three-way microvalves and six actuation micropumps, as shown in Fig. 2. The area of the three-way microvalve is about 8.5 $\times$ 4.2 mm [2], and the area of the actuation micropumps is about 7 $\times$ 7 mm [24]. In addition, each micropump must be connected to a flow sensor to measure the flow rate [21]. Five independent fluid-flow cycles dramatically increase the complexity of the system design and fabrication. Based on the basic design size for each processor: mixer, closed-chamber PCR, detector, and purifier, the size S of the flow-control devices for the continuous-flow PCR is

$$S \geq 10 \times (8.5 \times 4.2) + 6 \times (7 \times 7) \simeq 700 \text{ mm}^2.$$

However, the droplet-based PCR system requires smaller and more convenient electronic control components. The fluidic flow is easy to control. Because the dimension of electrodes ranges from hundreds of micrometers to one millimeter, the size of flow-control devices for the droplet-based PCR system is around several square millimeters.

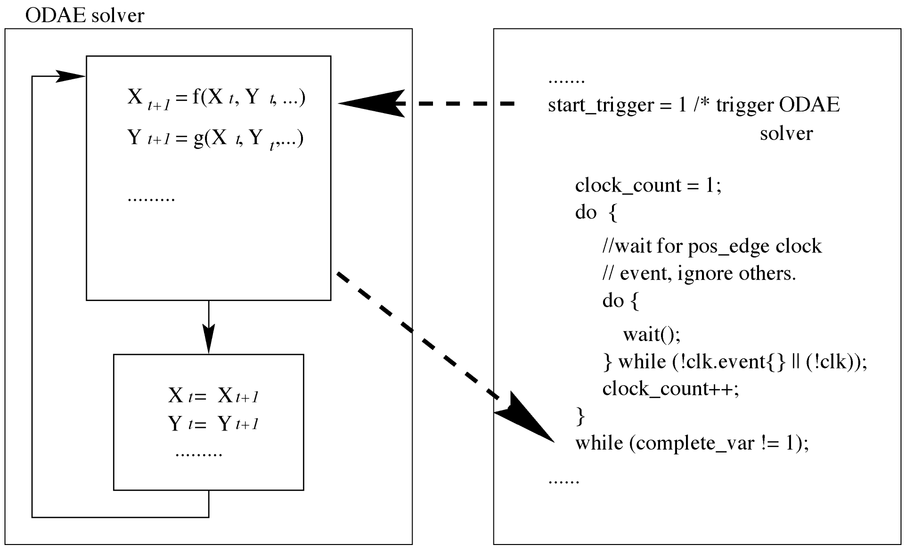
C. Performance Evaluation

The performance evaluation of the system is useful in identifying how the design parameters affect the overall system performance, and it provides the guidance for system optimization. Due to the reconfigurable nature of the hierarchical MEFS architecture, and complex component behaviors represented with ODAEs, it is almost impossible to derive the analytical models to study the overall system performance. Hence, the numerical model and simulation are necessary. In the following sections, the system performance is evaluated using the performance-analysis metrics of system throughput, system-correction capability, and system-processing capability.

1) System Throughput: Fig. 10 and Table VIII show the system throughput comparison between the continuous-flow PCR system and the droplet-based PCR system. The slow transportation speed and the complex structure limits the reconfigurable continuous-flow system's wider application. The droplet-based system improves the system throughput, and also enhances the system yield.

2) System Correction Capability: Because of the limitations of UV absorbance and fluorescence in analysis [18], the backup detector using the conductivity technique may determine that the PCR product does not match the concentration requirement. In contrast to the continuous-flow PCR system, there are no limitations for droplet-flow movement. Therefore, if one of the PCR units is available, the unqualified liquid is sent back for another thermal cycling. System correction capability is an important measure of the system performance. Table IX compares the system-correction capacity between two systems when the interarrival time between fluidic samples is an exponential probabilistic distribution, the mean is 12 min. Droplet-based systems show very good correction capability, and higher yield.

The continuous-flow PCR system and the droplet-based PCR system have nearly the same percentages of unqualified fluidic samples after reaction: 12.2% and 14.4%, respectively.



```
.......
start_trigger = 1 /* trigger ODAE
                         solver

    clock_count = 1;
    do {
        //wait for pos_edge clock
        // event, ignore others.
        do {
            wait();
        } while (!clk.event{} || (!clk));
        clock_count++;
    }
    while (complete_var != 1);
......
```

Fig. 9. ODAE solver and process communication illustration.

TABLE VII
ELEMENTAL PARAMETERS AND INITIAL NOMINAL-DESIGN VALUES

Parameters	Values	Units
Length of the cantilever (L)	1600	μm
Width of the cantilever(b')	1000	μm
Thickness of the cantilever (h')	15	μm
Height of the valve seat (h)	50	μm
Length of the valve seat (l_1)	5	μm
Width of the valve seat (b)	400	μm
Length of the cantilever over valve seat(l_2)	100	μm
Young's Modulus (E)	146.9	GPa
Air Pressure(P_a)	100000	P_a

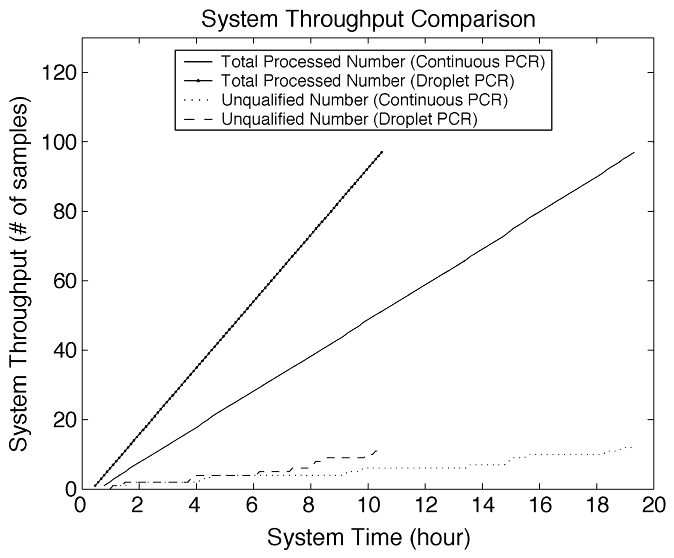


Fig. 10. System throughput comparison between the continuous-flow PCR system and the droplet-based PCR system. The droplet PCR system shows higher system throughput.

The continuous-flow PCR system does not have the correction capability. However, in the droplet-based PCR system, 10 out of 13 unqualified DNA solutions go back for one more thermal cycling, and 8 out of 10 unqualified fluidic samples are corrected after reprocessing. The correction percentage of

TABLE VIII
SYSTEM-THROUGHPUT COMPARISON WITH 100 DNA SOLUTIONS

PCR system	Finished time
Reconfigurable continuous-flow PCR	19.31hrs
Droplet-based PCR	10.48hrs

the droplet-based PCR is 61.5%, and the final percentage of unqualified fluidic samples is reduced to 5.5%.

3) System-Processing Capacity: Acquisition rate (workload) is another important system-level design parameter influencing system performance. For a given architecture, the microfluidic system possesses a saturation-processing capacity where resources are maximally utilized. Workloads less than saturation capacity under-utilize resources, whereas workloads greater than saturation capacity may decrease system quality. For instance, incoming fluidic samples have to wait longer if the system is saturated.

We assumed previously that the liquid sample acquisition rate is modeled by an exponential probabilistic distribution. Fig. 11 shows the system processing capacity of the continuous-flow PCR system and the droplet-based PCR system, respectively. The acquisition rate ranges from 1/700 to 1/50.

Fig. 11 shows the system processing capability versus different traffic rates. The vertical axis presents the system processing capability, denoted by using the number of processed fluidic samples per hour. The horizontal axis represents the sample traffic rate, λ, meaning the number of fluidic sample arriving per second. When the sample traffic rate is low, the system throughput is nearly linear; the performance of the system approximates is ideal. At increased input rates, i.e., reduced interarrival time, the actual system-processing capability increases and soon reaches saturation. Fig. 11 shows that the droplet-based PCR system has higher processing capacity.

VIII. CONCLUSION

We have demonstrated how MEFS can be modeled using SystemC. We have presented a performance comparison between

TABLE IX
SYSTEM-CORRECTION CAPACITY

	Continuous-flow PCR	Droplet-based PCR
Total number of processed samples	90	90
Number of internal unqualified samples	11	13
Number of post-process samples	0	10
Number of qualified sample after post-process	0	8
Number of final unqualified samples	11	5
Unqualified sample percentage	12.2%	5.5 %
Correction Percentage	0 %	61.5%

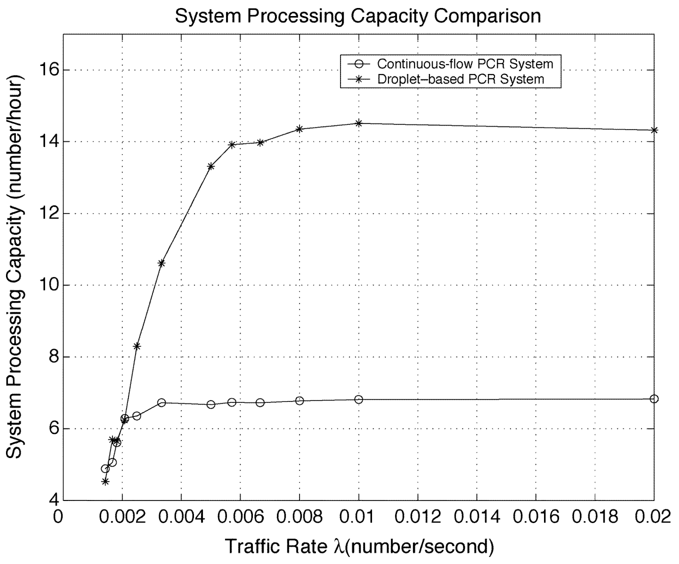


Fig. 11. System processing capability versus different traffic rate λ. After the system reaches the saturation, system processing capability (throughput) remains constant regardless of input rate variation.

two types of microfluidic systems: continuous-flow systems and droplet-based systems. The comparison is based on a special microfluidic application—a PCR system. The modeling and simulation of PCR systems are based on the SystemC design environment. The performance comparison includes the system throughput, system-correction capacity, system-processing capacity, and system-design complexity. We have demonstrated that the droplet-based microfluidic system provides higher performance, as well as lower design and integration complexity.

REFERENCES

[1] Semiconductor Industry Association. (2001) International Technology Roadmap for Semiconductors (ITRS). [Online] Available: http://public.itrs.net.

[2] T. Ohori, S. Shoji, K. Miura, and A. Yotsumoto, "Partly disposable three-way microvalve for a medical micro total analysis system (TAS)," *Sensors Actuators A*, vol. 64, no. 1, pp. 57–62, 1998.

[3] T. Bourouina, "Design and simulation of an electrostatic micropump for drug-delivery application," *J. Micromech. Microeng.*, vol. 7, no. 3, pp. 186–188, 1997.

[4] J. Pfahler, J. Harley, and H. Bau, "Liquid transport in micro and submicro channels," *Sensors Actuators A*, vol. 22, 1990.

[5] M. Pollack, R. B. Fair, and A. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications," *Appl. Phys. Lett.*, vol. 77, no. 11, pp. 1725–1726, 2000.

[6] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 1463–1468, Dec. 2001.

[7] P. Y. Chiou , M. C. Wu, H. Moon, C. J. Kim, and H. Toshiyoshi, "Optical actuation of microfluidics based on opto-electrowetting," in *Proc. Solid-State Sensor, Actuator, Microsyst. Workshop*, Hilton Head Island, SC, June 2–6, 2002, pp. 269–273.

[8] L. Stryer, *Biochemistry*, 4th ed. New York: W. H. Freeman, 1995.

[9] H. Swerdlow, B. J. Jones, and C. T. Wittwer, "Fully automated DNA reaction and analysis in a fluidic capillary instrument," *Anal. Chem.*, vol. 69, no. 5, pp. 848–855, 1997.

[10] G. Arnout, "SystemC standard," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 573–577.

[11] D. Ramanathan, R. Roth, and R. Gupta, "Interfacing hardware and software using C++ class libraries," in *Proc. Int. Conf. Comput. Design*, 2000, pp. 445–450.

[12] S. Y. Liao, "Toward a new standard for system-level design," in *Proc. 8th Int. Workshop Hardware/Software Codesign*, 2000, pp. 2–6.

[13] H. J. Schlebusch, "SystemC based hardware synthesis becomes reality," in *Proc. 26th Int. Euromicro Conf.*, vol. 1, 2000, p. 434.

[14] T. Zhang, K. Chakrabarty, and R. B. Fair, "Integrated hierarchical design of microelectrofluidic systems using SystemC," *Microelectron. J.*, vol. 33, pp. 459–470, 2002.

[15] A. Dewey and R. B. Fair, "Toward microfluidic system (MEFS) computing and architecture," in *Proc. 3rd Int. Conf. Modeling Simulation Microsyst.*, 2000, pp. 142–145.

[16] S. Shoji, "Microfabrication technologies and micro flow devices for chemical and bio-chemical micro flow systems," in *Proc. Int. Conf. Microprocess. Nanotechnol.*, 1999, pp. 72–73.

[17] T. Zhang, K. Chakrabarty, and R. B. Fair, *Microelectrofluidic Systems: Modeling and Simulation*. Boca Raton, FL: CRC, 2002.

[18] C. G. J. Schabmueller, M. A. Lee, A. G. R. Evans, A. Brunnschweiler, G. J. Ensell, and D. L. Leslie, "Closed chamber PCR chips for DNA amplification," *Eng. Sci. Educ. J.*, vol. 9, no. 6, pp. 259–264, 2000.

[19] S. McWhorter and S. A. Soper, "Conductivity detection of polymerase chain reaction products separated by micro-reversed-phase liquid chromatography," *J. Chromatog. A*, vol. 883, no. 1–2, pp. 1–9, 2000.

[20] T. S. J. Lammerink, V. L. Spiering, M. Elwenspoek, J. H. J. Fruitman, and A. van den Berg, "Modular concept for fluid handling systems: A demonstrator micro analysis system," in *Proc. IEEE 9th Annu. Int. Workshop Micro Electro Mech. Syst.*, 1996, pp. 389–394.

[21] A. Rasmussen and M. E. Zaghloul, "The design and fabrication of microfluidic flow sensors," in *Proc. Int. Symp. Circuits Syst.*, vol. 5, 1999, pp. 136–139.

[22] N. Schwesinger, T. Frank, and H. Wurmus, "A modular microfluid system with an integrated micromixer," *J. Micromech. Microeng.*, vol. 6, pp. 99–102, 1996.

[23] M. U. Kopp, A. J. de Mello, and A. Manz, "Chemical amplification: continuous-flow PCR on a chip," *Science*, vol. 280, pp. 1046–1048, 1998.

[24] R. Zengerle, S. Kluge, M. Richter, and A. Richter, "A bidirectional silicon micropump," *Sensors Actuators A*, vol. 50, pp. 81–86, 1995.

[25] J. Ulrich and R. Zengerle, "Static and dynamic flow simulation of a KOH-etched microvalve using the finite-element method," *Sensors Actuators A*, vol. 53, pp. 379–385, 1996.

[26] M. Washizu, "Electrostatic actuation of liquid droplets for micro-reactor applications," in *Proc. IEEE Industry Applicat. Soc. Annu. Meeting*, 1997, pp. 1867–1873.

[27] J. Lee and C.-J. Kim, "Surface-tension-driven microactuation based on continuous electrowetting," *J. Microelectromech. Syst.*, vol. 2, no. 9, pp. 171–180, 2000.

[28] E. O. Doebelin, *System Dynamics: Modeling, Analysis, Simulation, Design*. New York: Marcel Dekker, 1998.

[29] D. Rowell and D. N. Wormley, *System Dynamics: An Introduction*. Upper Saddle River, NJ: Prentice Hall, 1997.

[30] H. V. Vu and R. S. Esfandiari, *Dynamic Systems: Modeling and Analysis*. New York: McGraw-Hill, 1997.

[31] T. Zhang, F. Cao, A. Dewey, R. B. Fair, and K. Chakrabarty, "Performance analysis for microelectrofluidic system using hierarchical modeling and simulation," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 482–491, May 2001.

[32] T. Zhang, A. Dewey, and R. B. Fair, "A hierarchical approach to stochastic discrete and continuous performance simulation using composable software components," *Microelectron. J.*, vol. 31, no. 1, pp. 95–104, 2000.

[33] A. A. B. Pritsker, *Simulation With Visual SLAM and AweSim*. New York: Wiley, 1997.

[34] C. Wohlin, C. Nyberg, and A. Larsson, "Reusable simulation models for performance analysis of intelligent networks," in *Proc. Conf. Intell. Networks New Technol.*, 1996, pp. 143–154.

[35] G. K. Fedder and Q. Jing, "A hierarchical circuit-level design methodology for microelectromechanical systems," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 1309–1315, Oct. 1999.

[36] M. Orshansky, J. Chen, and H. Chenming, "A statistical performance simulation methodology for VLSI circuits," in *Proc. Design Automation Conf.*, 1998, pp. 402–407.

[37] P. Voigt, G. Schrag, and G. Wachutka, "Microfluidic system modeling using VHDL-AMS and circuit simulation," *Microelectron. J.*, vol. 29, no. 11, pp. 791–797, 1998.

[38] Coventor Inc. Conventroware: Architect: behavior models. [Online] Available: http://www.coventor.com/coventorware/architect/behavior-al_models.html.

[39] R. L. Burden, *Numerical Analysis*. Boston, MA: Prindle, Weber and Schmidt, 1985.

[40] T. Zhang, K. Chakrabarty, and R. B. Fair, "Design of reconfigurable composite microsystems based on hardware/software co-design principles," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 987–995, Aug. 2002.

[41] A. Dewey, H. Ren, and T. Zhang, "Behavior modeling of microelectromechanical systems (MEMS) with statistical performance variability reduction and sensitivity analysis," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 105–113, Feb. 2000.

[42] Z. Mrcarica, V. B. Litovski, M. Jakovljevic, and H. Detter, "Hierarchical modeling of microsystems in an object-oriented hardware description language," in *Proc. 21st Int. Conf. Microelectron.*, 1997, pp. 14–17.

Tianhao Zhang (S'98–M'01) received the B.E. and M.S.E. degrees from Tsinghua University, Beijing, China, in 1992 and 1997, respectively, and the Ph.D. degree from Duke University, Durham, NC, in 2001, all in computer science and engineering.

He is now a Member of the CAD Consulting Staff, Research and Development Department, Cadence Design System, Inc., Cary, NC. He is the author of *Microelectrofluidic Systems: Modeling and Simulation* (Boca Raton, FL: CRC Press, 2002), and tens of papers in journals and refereed conference proceedings. His current research interests include computer-aided design, VLSI design, MEFS modeling and simulation.

Dr. Zhang is a Member of Sigma Xi. He serves as a Reviewer for IEEE/ACM Design Automation Conference.

Krishnendu Chakrabarty (S'92–M'96–SM'00) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in computer science and engineering.

He is currently an Associate Professor of Electrical and Computer Engineering at Duke University, Durham, NC. From 2000 to 2002, he was also a Mercator Visiting Professor at the University of Potsdam, Potsdam, Germany. His current research projects include design and testing of system-on-a-chip integrated circuits, embedded real-time systems, distributed sensor networks, modeling, simulation, and optimization of microelectrofluidic systems, microfluidics-based chip cooling. He is a coauthor of two books: *Microelectrofluidic Systems: Modeling and Simulation* (Boca Raton, FL: CRC Press, 2002) and *Test Resource Partitioning for System-on-a-Chip* (Norwell, MA: Kluwer, 2002), and an editor of *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation* (Norwell, MA: Kluwer, 2002). He has published over 150 papers in journals and refereed conference proceedings, and holds a US patent in built-in self-test.

Dr. Chakrabarty is a Member of ACM, ACM SIGDA, and Sigma Xi. He is a recipient of the National Science Foundation Early Faculty (CAREER) Award and the Office of Naval Research Young Investigator Award. He received a best paper award at the 2001 Design, Automation, and Test in Europe (DATE) Conference. He is also the recipient of the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany. He is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, an Editor of the *Journal of Electronic Testing: Theory and Applications (JETTA)*, and a member of the editorial board for *Sensor Letters* and the *Journal of Embedded Computing*. He has also served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: ANALOG AND DIGITAL SIGNAL PROCESSING. He serves as Vice Chair of Technical Activities of the IEEE's Test Technology Technical Council and is a Member of the program committees of several IEEE/ACM conferences and workshops.

Richard B. Fair (S'63–M'75–SM'75–F'90) received the Ph.D. degree from Duke University, Durham, NC, in 1969.

He is a Professor of Electrical and Computer Engineering at Duke University. He spent 12 years at Bell Labs working on semiconductor devices and integrated circuit technology. He returned to North Carolina in 1981 and spent 13 years as a Vice President of the Microelectronics Center of North Carolina, having responsibilities in chip design, computer-aided design, packaging, process technology, and MEMS. His research group first demonstrated microdroplet transport over electrode arrays based on electrowetting actuation. He has published over 140 papers in refereed journals and conference proceedings, written 10 book chapters, edited eight books or conference proceedings, and given over 100 invited talks, mostly in the area of semiconductor devices or the fabrication thereof. His current research interests include bio-MEMS and digital microfluidic chips.

Prof. Fair is a Fellow of the Electrochemical Society. He is also a past Editor-in-Chief of the PROCEEDINGS OF THE IEEE, and he has served as an Associate Editor of the IEEE TRANSACTIONS ON ELECTRON DEVICES. He was a recipient of the IEEE Third Millennium Medal in 2000 and the 2003 Solid State Science and Technology Prize and Medal from the Electrochemical Society.