ELSEVIER

Microelectronics Journal 33 (2002) 459–470

Microelectronics Journal

www.elsevier.com/locate/mejo

Integrated hierarchical design of microelectrofluidic systems using SystemC

Tianhao Zhang[a,*], Krishnendu Chakrabarty[b], Richard B. Fair[b]

[a]*Cadence Design Systems, Inc, Suite 130, 200 Regency Forest Dr. Cary, NC 27511, USA*
[b]*Department of Electrical and Computer Engineering, P.O. Box 90291, Duke University, Durham, NC 27708-90291, USA*

Received 7 August 2001; accepted 9 November 2001

Abstract

This paper describes the role of SystemC in developing an integrated modeling and simulation environment for microelectrofluidic systems (MEFS). Based on the unique modeling and simulation needs for MEFS, we examine suitability of several existing simulation languages. These languages include VHDL/VHDL-AMS, SLAM, Matlab, C/C++, and SystemC. Next, SystemC is justified as a viable candidate for complete MEFS modeling and simulation. The architecture of the environment and the associated functional packages are discussed. Its application is illustrated through the design of a microchemical handling system.

Keywords: SystemC; Microsystem; System-level

1. Introduction

Composite microsystems that incorporate microelectromechanical and microelectrofluidic devices are emerging as the next generation of system-on-a-chip (SOC) designs. These systems combine microstructures with solid-state electronics to integrate multiple energy domains. Microelectrofluidic systems (MEFS) is an area of research that addresses the miniaturization of composite devices and systems, and the study of new applications associated with the handling of liquids and gases. Microfluidics not only offers the obvious advantage of size reduction, but it also reduces power dissipation and increases system reliability. Significant progress have been made towards the design of individual MEFS components. For example, the fluidic mixer [1], valves [2], have been designed, built, and studied extensively. Moreover, small systems that combine existing components into useful devices have been designed, and some of them have been built, i.e. DNA analysis devices [3].

Composite microsystem design is evolving into a multidisciplinary field requiring expertise in electrical, mechanical, and chemical engineering, as well as in computer science and manufacturing technology. This broad scope makes it nearly impossible to manually understand complicated factors influencing and limiting system performance. Thus system modeling and simulation are becoming important computer-aided-design tools for synthesis, analysis, and verification. These tasks often require the collaborative efforts of several teams and organizations, typically using different modeling languages and simulators.

Traditionally, several modeling languages and simulators are 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 [13]. On the other hand, logic modeling languages, for instance VHDL/VHDL-AMS, are used for low-level functional unit design [17]. 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.

SystemC is a new open source library in C++ [10]. It

* Corresponding author. Tel.: +1-919-481-6834; fax: +1-919-380-3020.
E-mail address: tzhang@cadence.com (T. Zhang).

0026-2692/02/$ - see front matter
PII: S0026-2692(01)00157-4

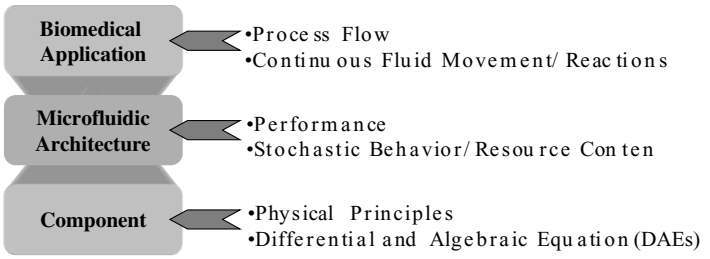

Fig. 1. Multi-layered modeling and simulation hierarchy for MEFS.

supports hardware–software co-design and the description of the architecture of complex systems consisting of both hardware and software components. It has been used in electronic hardware/software co-design [4], system-level design [9], and hardware synthesis [5,16]. In this paper, we introduce a new application area for SystemC: the design of MEFS. We evaluate the suitability of SystemC for MEFS hierarchical design, and propose a MEFS hierarchical modeling and simulation environment based on SystemC. The main contributions of the paper are summarized below:

- We define the basic variables and elements needed to describe MEFS behavior from the lower component level to the architectural and biomedical/chemical application levels.
- Based on the special needs for MEFS hierarchical behavior, we compare the suitability of several simulation languages for hierarchical design. These languages include VHDL/VHDL-AMS, SLAM, Matlab, C/C++, and SystemC.
- We propose a hierarchical integrated design environment for MEFS with SystemC. The architecture of the environment and the associated functional packages are discussed.
- We illustrated this environment by designing a special MEF system: a microchemical handling system (MCHS).

To the best of our knowledge, this is the first attempt to develop a comprehensive integrated MEFS design environment with SystemC [6]. The organization of this paper is as follows. The basic variables and elements needed to describe MEFS behavior are defined in Section 2. In Section 3, the suitability of several simulation languages for MEFS hierarchical design is evaluated, and SystemC is proposed as a viable candidate. In Section 4, a hierarchical modeling and simulation environment with SystemC is presented. The architecture of the environment and the associated functional packages are discussed. In addition, its application to the system design of a MCHS is illustrated in Section 5. Finally, conclusions are presented in Section 6.

2. MEFS modeling and simulation characteristics

In this section, we present the basic hierarchical architecture and functional execution of MEFS. In addition, we define fundamental variables and elements needed to describe MEFS from the lower component level to the higher system-level. These fundamental variables capture system behavior, and they are critical parts of a MEFS modeling and simulation languages.

2.1. Hierarchical perspective

Fig. 1 illustrates the MEFS hierarchical perspective. Each layer of abstraction presents unique model fidelity, domain representation, and simulation efficiency requirements and challenges.

The biomedical application layer is the highest level of abstraction. It involves process flow modeling, as well as the simulation of continuous fluid movement and bio/chemical reactions directed towards an application, such as microdialysis, chemotherapy, genetic analysis, or cell filtration. Evaluating biomedical applications requires lower level information, such as channel pressure drops and pump throughput, which are associated with the second level of abstraction—the reconfigurable microliquid handling architecture. This level of abstraction involves performance modeling and simulation of the stochastic behavior of the major component, resources and their aggregate operation in executing a biomedical application. Evaluating architectural performance involves, in turn, lower level information, such as microfluidic transport and the device operation, which are associated with the lowest level of abstraction—microfluidic components. This level of abstraction involves detailed circuit simulation requiring integro—differential and algebraic equations, which characterize physical properties and processes. Here, the microfluidic component layer of abstraction is defined as the circuit-level, while the biomedical/chemical application and reconfigurable MEFS architecture are defined as the system-level.

2.2. Dynamic modeling and simulation at circuit-level

Direct numerical simulation of the dynamic behavior of MEFS devices requires three-dimensional, distributed nonlinear models. These models are difficult to develop and computationally expensive to use. Therefore, it is necessary to decrease the degrees of freedom by reducing meshed three-dimensional device models to two-dimensional circuit-level lumped-element models. These models present

the overall component behavior using ordinary differential and algebraic equations (ODAEs). The use of lumped-element models for microsystem devices results in compact and efficient representations. It also avoids potential convergence problems due to the coupling of heterogeneous simulators. Therefore, a single modeling and simulation language should be able to represent the fundamental physical principles of different energy domains. This description capacity spans disparate energy domains and allows simultaneous consideration of the multiple physical phenomena in solving overall system behavior with ODAEs.

2.3. System-level modeling and simulation

MEFS system-level modeling includes architectural stochastic modeling and biomedical/chemical process flow modeling. We need to describe different behavioral perspectives, for instance the queuing nature of biochemical DNA analysis systems, continuous thermal reactions, and liquid sample arrival events. Three representations, named 'world-views', (i) discrete event-scheduling, (ii) discrete process-interaction, and (iii) continuous, are all necessary for simulation languages [13]. In addition, the fluidic-sample-oriented MEFS behavior requires a complex but flexible data structure. Dynamic data structures are also required to capture the runtime non-determinism of stochastic systems. Moreover, a variety of constructs and statements supporting both sequential and concurrent executional semantics are required.

Time-advanced mechanisms require simulation languages to support a simulation clock that can give the current value of simulated time. It is also necessary to synchronize events in the simulation so that parallel events can be properly modeled by the simulator on a sequential computer. In addition, a statistical analysis approach is required for stochastic systems so that various usage information can be compiled during system execution. Probabilistic and statistical analysis require multiple data types, powerful mathematical resources (function libraries), and operating system storage (file) input/output.

3. Suitability evaluation

3.1. VHDL-AMS for circuit-level modeling and simulation

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 [15]. VHDL-AMS supports circuit-level modeling and simulation of continuous and discrete systems with conservative and non-conservative 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. However, the processor-oriented modeling perspective of VHDL-AMS limits its applicability for MEFS fluidic-sample-oriented analysis. For instance, it cannot provide a complex and flexible data structure to describe the fluidic sample characteristics.

3.2. VHDL for system-level modeling and simulation

VHDL supports a broad range of constructs to describe detailed logic and abstract algorithms. It also provides both concurrent execution semantics and sequential execution semantics for parallelism, and for ordering procedures and functions. However, VHDL's event-driven perspective does not directly possess the capability to represent the object-oriented and the queuing nature of MEFS higher level stochastic behavior. In addition, the language syntax prevents the wider application of VHDL to system-level modeling [18]. Although the VHDL'92 version adds shared variables to enhance the VHDL's system-level design capability, the shared variable's constrained scope limits its application from the scalable system design objective, and there is still some controversy about the rationale of shared variable. Some research groups suggest extending the VHDL language syntax for system-level modeling [14]. However, these extensions focus more on the electrical energy domain, and they can potentially destroy the integrity of VHDL. Hence, it is now well accepted that existing hardware description languages cannot be effectively expanded to support system-level modeling and simulation [10].

3.3. Performance language—SLAM

SLAM is a high-level performance modeling language [13]. It provides the capacity to describe the overall system as a stochastic system. An important aspect of SLAM is that alternate world-views 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.

3.4. C/C++ and Matlab

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.

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. It is very hard to study MEFS stochastic behavior for bio/chemical applications by using Matlab. In addition, Matlab does not directly support coupled-energy domain modeling. It lacks the capacity for discrete event-driven modeling and concurrent simulation, and does not support multiple logical values. Moreover, there is general consensus among most Matlab users that certain Matlab programs run extremely slowly.

3.5. SystemC

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. To model and simulate continuous world-view with SystemC, differential equations with respect to time can be discretized and transformed into corresponding difference equations.

Due to the complexity of MEFS designs, it is necessary 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. The only obvious 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.

In summary, after evaluating the suitability of these languages for MEFS hierarchical design, VHDL/VHDL-AMS, SLAM II, C/C++, and Matlab are not suitable to handle the complete MEFS modeling and simulation. In contrast, SystemC is a viable candidate to develop a MEFS hierarchical modeling and simulation environment. This motivates the work reported in the rest of this paper.

4. SystemC design environment

On the analogy of the Gasjki and Kuhn's Y-chart in microelectronics CAD [11], a MEFS closed-loop integration design environment should extend system design from the component level to the system-level, and include the following two different functional packages: system-level modeling and simulation package, circuit-level component modeling and simulation package. These functional packages are discussed in the following subsections.

4.1. System-level modeling package

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

4.1.1. MEFS behavioral description

Depending on the system-level characteristics of MEFS, the fundamental elements for system modeling include: (i) the storage part, which is used to temporarily store the fluidic samples, and examples of which includes fluidic input buffers and containment reservoirs; (ii) the transportation part, which is used to deliver fluidic samples from one site to another; (iii) 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: (iv) the timing clock to synchronize simulation events; and (v) a complex but flexible fluidic sample data structure. It contains the fluidic sample's physical properties, and simulation procedure records. These concepts will be illustrated in Section 6.

4.1.2. MEFS architecture 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. An example about MEFS architecture design will be illustrated in Section 6.

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.

4.2. Circuit-level component modeling package

The goal of MEFS component modeling and simulation is to study individual microfluidic components at the circuit-level of abstraction, emphasizing the definition of physical

```
-- ENERGY_SYSTEMS
signal <float>
     ENERGY,
     POWER ,
     PERIODICITY;

-- MECHANICAL_SYSTEMS
signal <float>
     TRANSLATION,
     FORCE,
     ROTATION,
     TORQUE;
```

```
-- ELECTRICAL_SYSTEMS
signal <float>
     VOLTAGE,
     CURRENT;

-- FLUIDIC_SYSTEMS
signal <float>
     PRESSURE,
     FLOW_RATE;

-- THERMAL_SYSTEMS
signal <float>
     TEMPERATURE,
     HEAT_FLOW;
```

Fig. 2. SystemC type declarations for composite microsystems.

properties and their relationships across multiple energy domains.

4.2.1. Energy domain behavior declarations

Based on the microfluidic component modeling common issues discussed in Section 2.2, coupled-energy component modeling requires the declaration of specific variables to represent individual energy domains and disciplines. Similar to the energy declaration in VHDL-AMS, Fig. 2 shows the same declarations for the variables for each energy domains in SystemC. These declarations use the *signal* construct of SystemC.

4.2.2. 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 users' 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. Therefore, users have to code various DAEs solvers with SystemC and add them into a SystemC component behavior model.

5. Case study: microchemical handling system

Fig. 3 shows a representative MCHS. This design is based on a reconfigurable microfluidic system architecture being developed at Duke university [12]. This design makes use of several basic microfluidic component [1].

The architecture of MCHS is composed with processing elements and the reconfigurable mother-board. The reconfigurable mother-board contains the liquid entrances, a bus storage buffer containing n equal volume cells, and a single bus channel connecting the bus storage buffer to the exit. Several standard I/O interface ports are located along the bus channel to connect the processing elements. A standard processing element has a standard I/O interface port, a bidirectional micropump providing pressure-driven flow throughout the system, and a reaction chamber involving mixers, analyzers, and catalyzers. Liquid samples enter

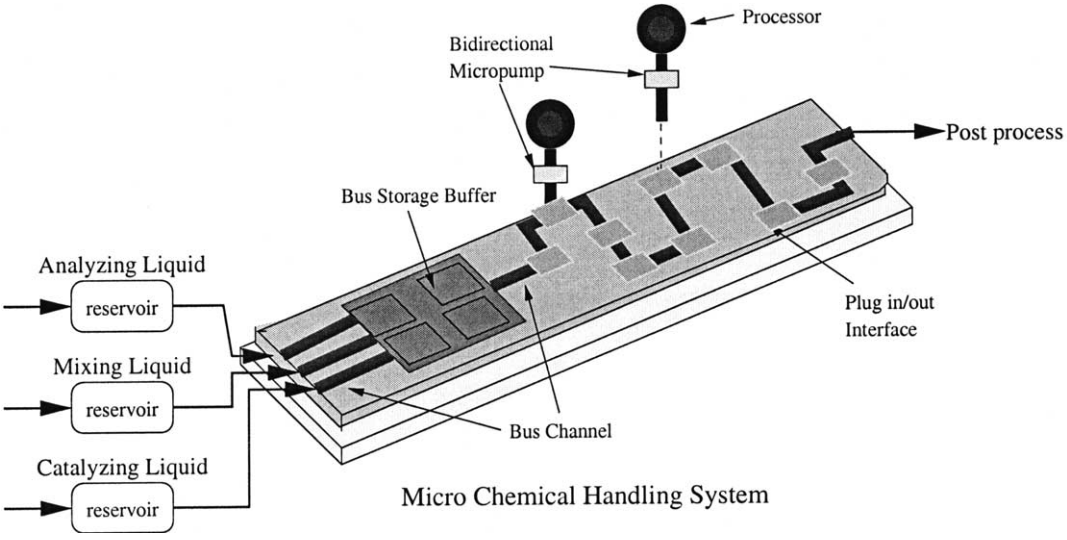

Fig. 3. A representative microchemical handling system based on basic microfluid components.

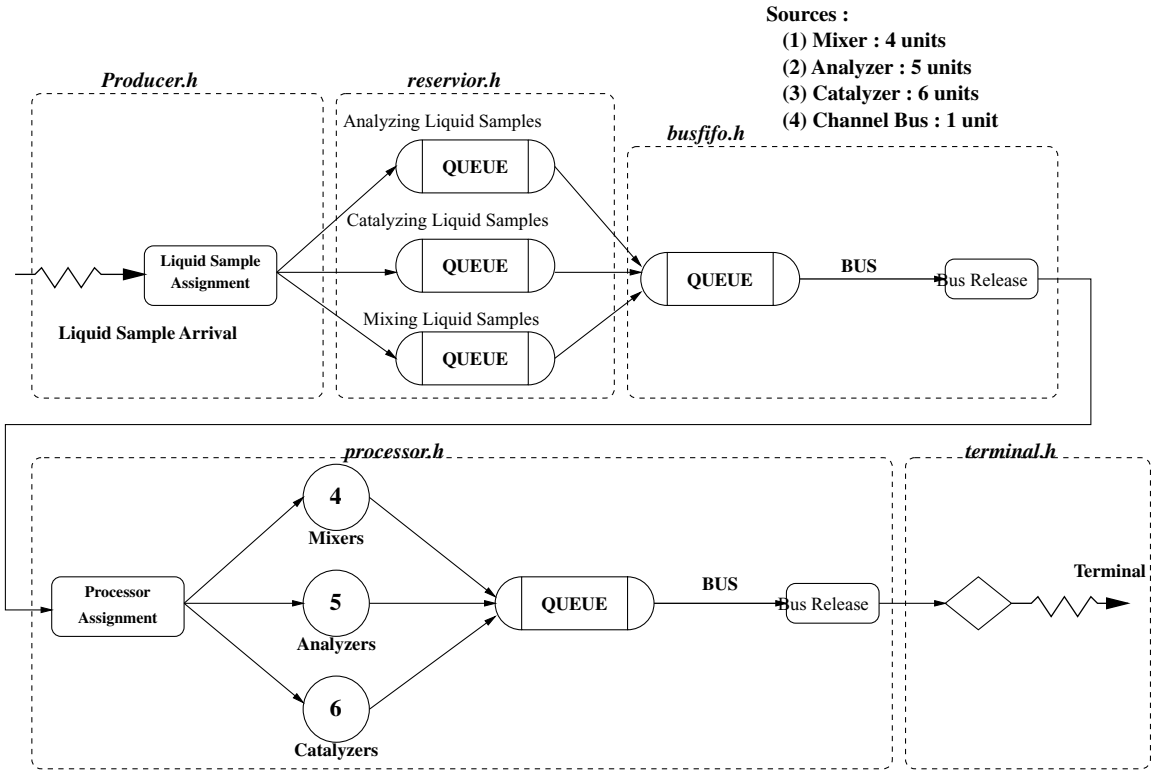

Fig. 4. Schematic network modeling of the microchemical handling system based on a queuing network model.

the system from three containment reservoirs. Samples entering the system are queued in a bus storage buffer. Micropumps intermittently draw liquid samples from the bus storage buffer to individual processors when the bus channel and related processor are available. After processing, the sample is pumped from the processor to the output, again when the bus channel is available. Due to the time spent transporting a liquid sample into a processing element, executing the chemical reaction, and transporting the resulting sample out of the system, chemical liquid samples initially input into a bus storage buffer until applicable resources are available.

5.1. System modeling

The architectural simulation model of the MCHS is developed using a combination of process, event, and continuous control paradigms. Operation of the MCHS is denoted by the flow of entities (clients) through a network structure consisting of nodes and branches denoting resources, queues for resources, activities, and entity flow decisions.

Fig. 4 shows a diagram of the queuing network model of the MCHS. This model is built based on the SystemC design environment. The chemical handling process can be separated into five stages, depending on the liquid-process routine. The first stage—the initial sample creation, is coded in *producer.h*. The second stage is initial sample acquisition-fluidic sample entering and being stored in the appropriate containment reservoir. This stage is coded with *reservoir.h*. In stage three, fluidic samples are moved from containment reservoirs into the bus storage buffer that is not full. The behavior of this stage is coded in *busfifo.h*, then when the channel bus is free, samples are transported to the appropriate processor. The procedure—*processor.h*—is used for this stage. It consists of all processors. Fig. 5 shows its header and implementation code of one analyzer file. After processing, the processed liquid sample is transported from the outlet when the bus channel is available. This terminal stage is coded in *terminal.h*. Simulation results also are recorded in this stage for further data analyses.

Channel bus delivery time is dependent on the liquid flow rate, liquid sample volume, and the distance between locations. Processing duration is dependent on liquid sample properties and the type of reaction. Without loss of generality, the volume for each liquid sample is assumed to be the same (800 μl). Acquisition is modeled by a traffic of liquid samples separated by interarrival times, denoted by $\{T_1,T_2,\ldots\}$. These are independent, identically distributed (IID) random variables, and are characterized by an exponential probabilistic distribution given by (1) having a mean value of 15 s, that is $\lambda = 1/15$.

$$f(x) = \lambda e^{-\lambda x} \qquad x \leq 0 \tag{1}$$

Fig. 6 shows the program structure of this MCHS. Each function block is hierarchically connected to the toper

```
// header file:  processor.h
SC_MODULE (processor)
{// define interface
  sc_outmaster<int> fluid_select;
  sc_inmaster<fluid_type > fluid_in;
  sc_outmaster<fluid_type> fluid_out;
  sc_in_clk clk;
  processor_buffer<fluid_type>
     Analyzer, Mixer, Reactor, ...;
  void analyzer();
  void mixer();
  void reactor();
  SC_CTOR(processor)
  {
  SC_THREAD(analyzer);
   sensitive << clk;
   SC_THREAD(mixer);
   sensitive << clk;
   SC_THREAD(reactor);
   sensitive << clk;
  }
};
```

```
// implementation file:analyzer.cpp
void processor:analzyer()
 {
 while (true)
  {wait();
  fluid_select=analyzing;
  // occupy the channel
    •••
   item=fluid_in;
  //delivery
    •••
  //processing
    •••
  //delivery
    •••
   fluid_out = item;
  //release the analyzer
    •••
  //release the channel
    •••
   }
 }
```

Fig. 5. Program structure for processor.h and analyzer.cpp modules.

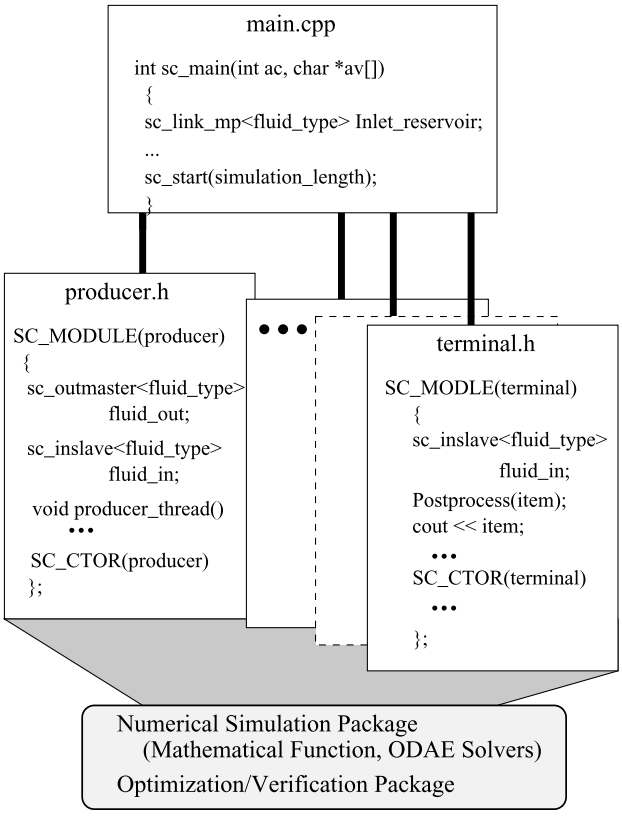

Fig. 6. Description of a microchemical handling system with SystemC.

level program. The connection between different functional blocks is defined on the toper level. Associated numerical simulation package and optimization package support the system modeling, simulation and optimization [7,8].

5.2. Thermal reaction process

The liquid thermal reaction process is very useful for various bio/chemical analysis, such as biochemical reactions with DNA [3]. The liquid and reagent are mixed in the reaction chamber and then heated. When the liquid reaches a certain temperature, the reaction process completes, and the liquid is pumped out. Let $y(t)$ and $x(t)$ denote the temperature of the heating stage (chamber) and the mixed liquid at time t, respectively. The initial temperature of the reaction chamber is assumed as $y(0) = 300\text{F}$. The functionality of reactor units is defined by the following differential equations. The temperature-changing rate of a fluidic sample is proportional to its temperature and the temperature of the surrounding medium, i.e.

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = 0.12 \times (300 - y(t))$$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = 0.04 \times (y(t) - x(t))$$

There are several numerical integration methods to solve the above ODAEs, here, the relaxation-based numerical integration techniques coded using SystemC are used. In order to improve the accuracy of the results computed, another simulation clock with higher frequency is used. Fig. 7 shows the integration ODAE result of a certain fluidic sample with the initial temperature at 54F.

5.3. Microvalve lumped-element nodal modeling

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 schematic view of the open valve is shown in Fig. 8 [2].

Where L is the length of the cantilever, l_1 is the length of the valve seat, l_2 is the length of the cantilever over valve seat, h' is the thickness of the cantilever, and h is the height of the valve seat. To derive an analytical result, the gap between the cantilever and the valve seat is divided into five pieces. Depending on the relation between the flow rate Φ and the different region pressures $\Delta p_i (i = 1, 2, \ldots, 5)$. The flow rate can be treated as a function of pressure difference p and the displacement y:

$$p = \sum_{i=1}^{V} \Delta p_i(\Phi, y) \tag{2}$$

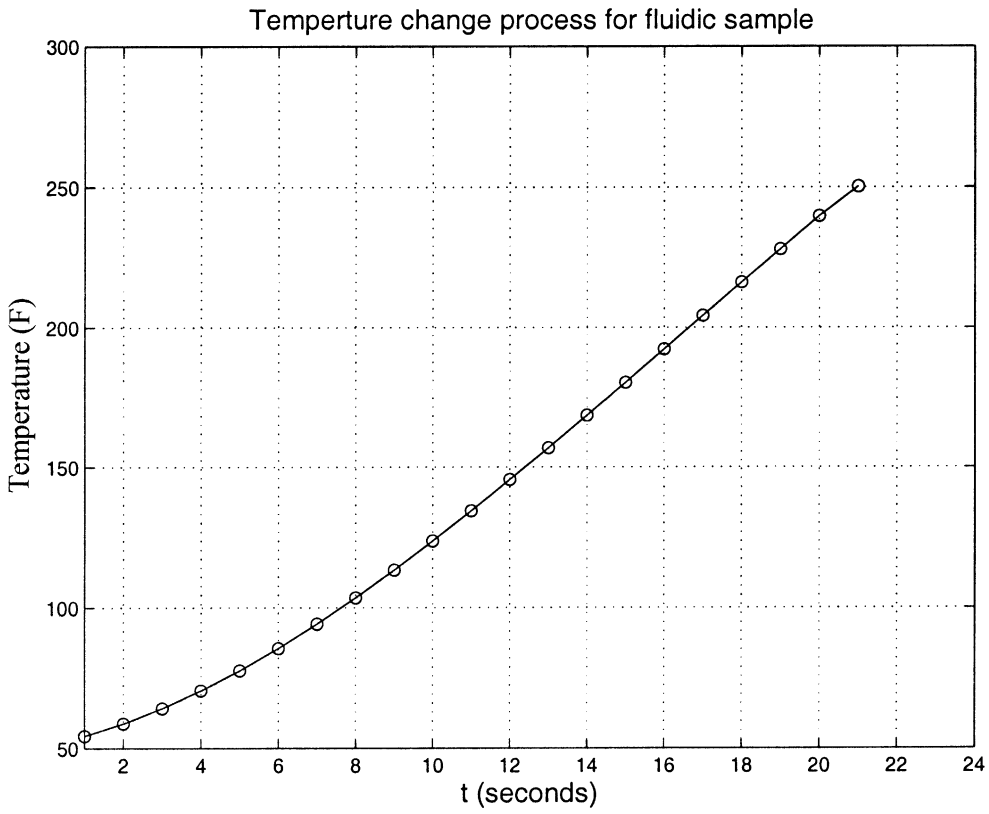

Fig. 7. Variation of temperature of a fluidic sample with time in the range of 54 to 250F.

The displacement y is determined by the pressure difference between the valves. The behavior of the cantilever could be described by a second-order differential Eq. (3)

$$m\ddot{y} + d\dot{y} + ky = pA \tag{3}$$

where m is effective mass of the cantilever, including the mass of cantilever and that of the liquid surrounding the cantilever, d, the damping constant determined by the geometry of the cantilever, and k, is the spring constant described by the geometry of the cantilever, and product materials. We build this microvalve analytical model with SystemC. Because these ODAEs are inherently non-linear and coupled, they can be solved only numerically. Fig. 9 shows the microvalve model coded by VHDL-AMS and SystemC, respectively. Because SystemC does not provide an associated simulator, the simultanenous ODAEs are combined with ODAE solver coded by user and solved by a Process: ODAEs();

Table 1 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. By setting the microvalve operating frequency at 100 Hz, the average flow rate of the microvalve can be calculated to be 5.86 ml/min.

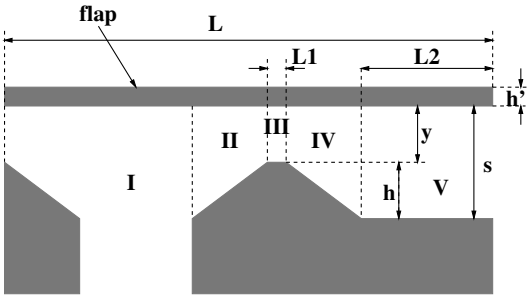

Fig. 8. Schematic view of the opening valve.

5.4. Performance analysis

Combining the system simulation with the stochastic

Table 1
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)	100,000	Pa

```
// 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. 9. Microvalve model coded by VHDL-AMS and SystemC.

macro model and component level simulation with lumped-element modeling discussed in earlier sections, hierarchical simulation results with SystemC are shown in Figs. 10–12. These simulation results show the system performance when there are 100 fluidic samples entering the system for processing. The performance analysis results include throughput, resource utilization, and execution time distributions. These data are useful in identifying how component performance metrics impact overall architectural performance, and they provide guidance for optimization.

The total time for each fluidic sample staying in the handling system consists of three periods: the waiting time for system resources, the processing time, and the microchannel delivering time. Fig. 10 shows the average values of three periods for each kind of fluidic samples with one-channel structure. The chemical fluidic sample waiting time occupies a too high proportion of the total cycle time. The channel bus is the principal bottleneck preventing liquid samples from accessing processors and processed liquid samples from being dispensed. The effects of increasing channel bus bandwidth by adding another parallel channel bus are studied in Figs. 11 and 12. Fig. 11 compares system throughput versus channel bus bandwidth and Fig. 12 compares system resource utilization versus channel bus bandwidth. The better architecture with the two-channel bus architecture reduces system processing time and increases resource availability.

Acquisition rate (workload) is another important system-level design parameter influencing system performance. For a given architecture, the MCHS has a saturation capacity, where resources are maximally utilized. Workloads less than saturation capacity under-utilizes resources, whereas workloads greater than saturation capacity may decrease system quality or even cause system failure. Thus, it is desirable to investigate saturation operating performance. Fig. 13 shows the MCHS performance under varying sample acquisition rates, λ changing from 1/100 to 1/5. The straight dash-point line shows an ideal case when availability of the system is guaranteed and all input samples are accepted and processed. The other curve is actual system performance. When the sample traffic rate is low, system throughput is nearly linear—the performance

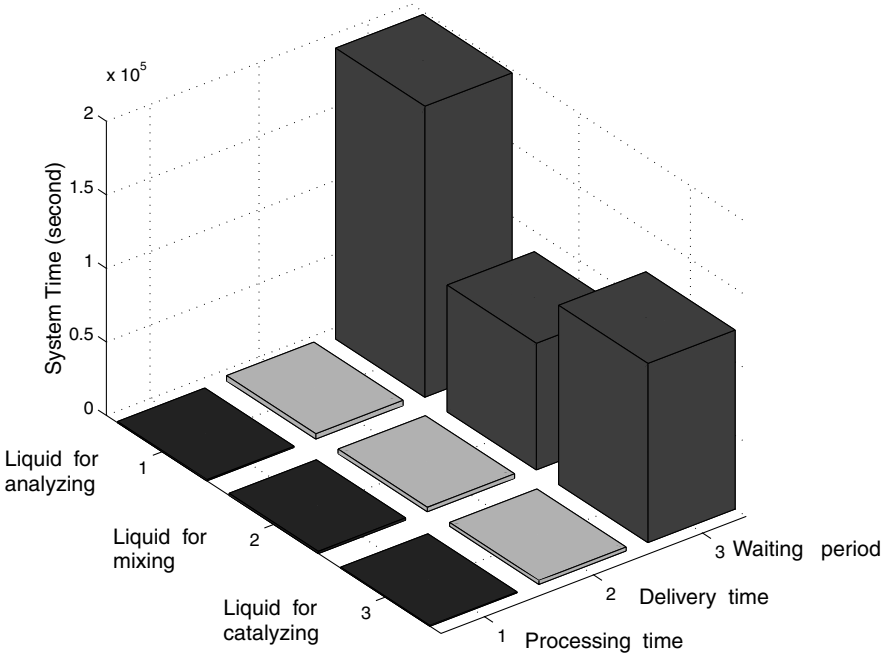

Fig. 10. Simulation results: average times for the different samples in various phases.

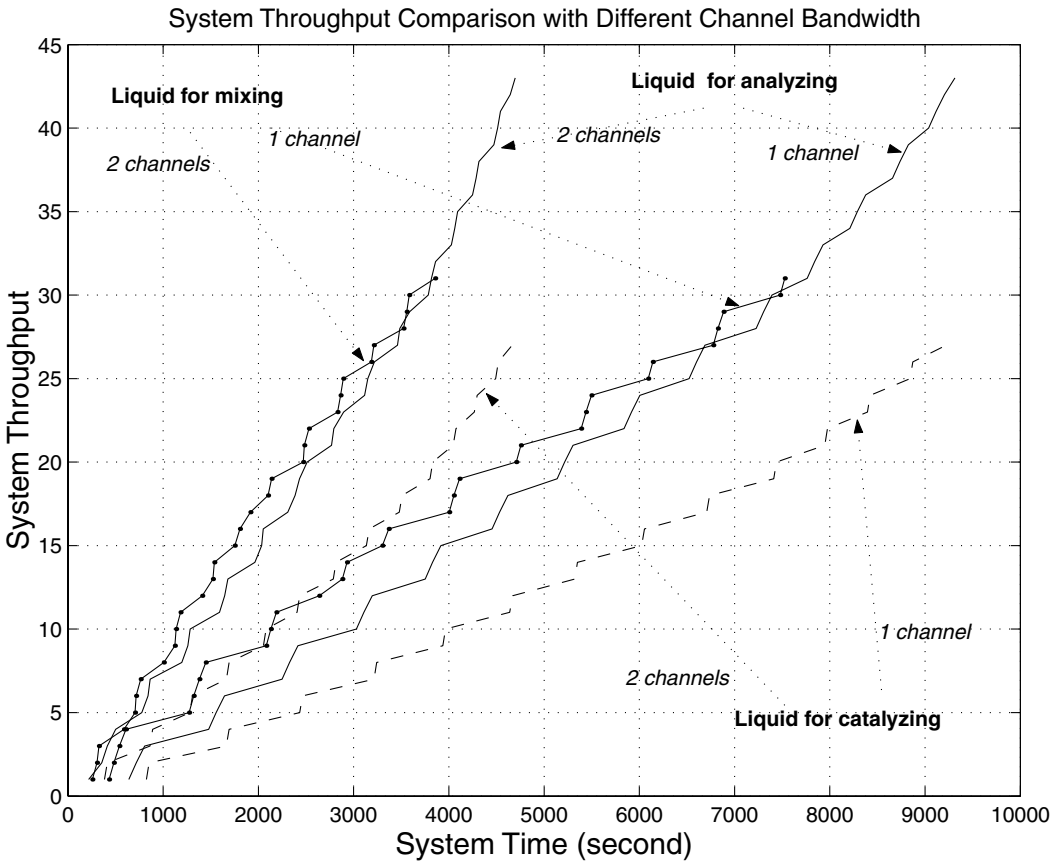

Fig. 11. Variation of system throughput with different channel bus bandwidths.

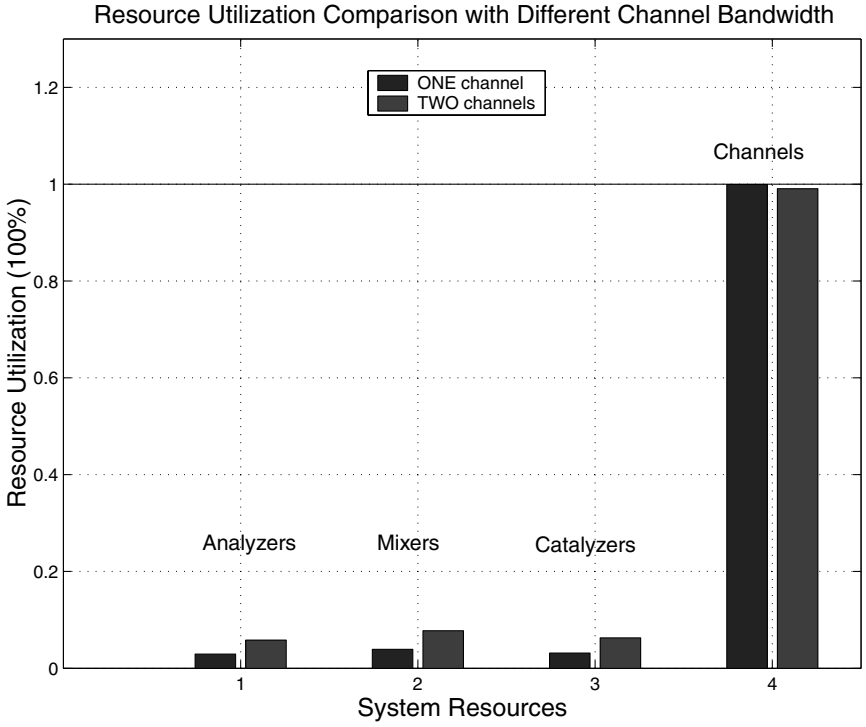

Fig. 12. System resource utilization.

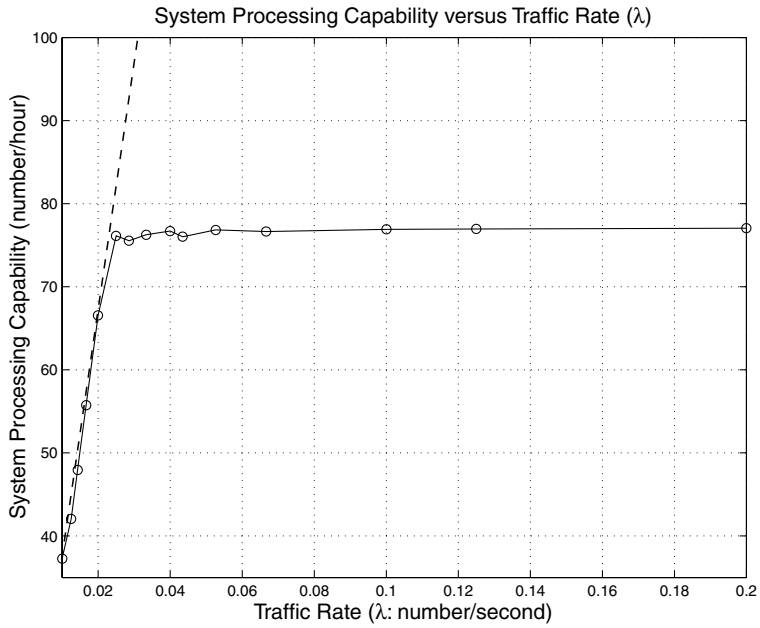

Fig. 13. Variation of system processing capability with change in traffic rate.

of the system approximates the ideal. At increased input rates, meaning reduced interarrival time, actual system processing capability increases and quickly reaches saturation.

6. Conclusion

n this paper, we demonstrated SystemC as a strong candidate for MEFS hierarchical modeling and simulation. The architectural features of a hierarchical modeling and simulation environment based on SystemC have been presented. This design environment extends from lower level component modeling and simulation to higher level system modeling and simulation. In addition, several functional packages have been described. The complete system modeling and simulation for a MCHS is presented depending on this design environment. Simulation results are presented and performance analyses are presented.

References

[1] N. Schwesinger, T. Frank, H. Wurmus, A modular microfluid system with an integrated micromixer, Journal of Micromechanics and Microengineering 6 (1) (1996) 99–102.

[2] J. Ulrich, R. Zengerle, Static and dynamic flow simulation of a KOH-etched microvalve using the finite-element method, Sensors and Actuators A53 (1996) 379–385.

[3] M.A. Burns, et al., An integrated nanoliter DNA analysis device, Science 282 (1998) 484–487.

[4] D. Ramanathan, R. Roth, R. Gupta, Interfacing hardware and software using C++ class libraries, Proceedings of the International Conference on Computer Design, 2000, pp. 445–450.

[5] H.J. Schlebusch, SystemC based hardware synthesis becomes reality, Proceedings of the 26th International Euromicro Conference, 1, 2000, pp. 434.

[6] T. Zhang, An integrated hierarchical modeling and simulation approach for microelectrofluidic systems, PhD Dissertation, Duke University, 2001.

[7] T. Zhang, K. Chakrabarty, R.B. Fair, Design of reconfigurable composite microsystems based on hardware/software co-design principles, Proceedings of the 4th International Conference Modeling and Simulation of Microsystems (MSM2001), Hilton Oceanfront Resort, Hilton Head Island, South Carolina, USA, 2001, pp. 148–152.

[8] A. Dewey, H. Ren, T. Zhang, Behavior modeling of microelectromechanical systems (MEMS) with statistical performance variability reduction and sensitivity analysis, IEEE Transaction on Circuit and Systems II: Analog and Digital Signal Processing 47 (2) (2000) 105–113.

[9] S.Y. Liao, Towards a new standard for system-level design, Proceedings of the 8th International Workshop on Hardware/Software Codesign, 2000, pp. 2–6.

[10] G. Arnout, SystemC standard, Proceedings of Asia and South Pacific Design Automation Conference, 2000, pp. 573–577.

[11] D.D. Gajski, R.H. Kuhn, Guest editor's introduction: new VLSI tools, IEEE Computer 16 (12) (1983) 4–8.

[12] A. Dewey, et al., Towards microfluidic system (MEFS) computing and architecture, Proceedings of the 3rd International Conference on Modeling and Simulation of Microsystems (MSM2000), San Diego, California, 2000, pp. 142–145.

[13] A. Alan, B. Pritsker, Simulation with Visual SLAM and AweSim, Wiley, New York, 1997.

[14] B. Djafri, J. Benzakki, OOVHDL: object oriented VHDL, Proceedings of VHDL International Users' Forum, 1997, pp. 54–59.

[15] A. Dewey, et al., VHDL-AMS Modeling Considerations and Styles for Composite Microsystems, Tutorial, DAC99, New Orleans, 1999.

[16] G. Economakos, et al., Behavioral synthesis with SystemC,

Proceedings of Conference on Design, Automation and Test in Europe, 2001, pp. 21–25.

[17] P. Voigt, G. Schrag, G. Wachutka, Microfluidic system modeling using VHDL-AMS and circuit simulation, Microelectronics Journal 29 (1998) 791–797.

[18] .T. Zhang, A. Dewey, R.B. Fair, A hierarchical approach to stochastic discrete and continuous performance simulation using composable software components, Microelectronics Journal 31 (2000) 95–104.