Microfluid Nanofluid (2013) 15:647–659
DOI 10.1007/s10404-013-1180-2

RESEARCH PAPER

Software-programmable continuous-flow multi-purpose lab-on-a-chip

Ahmed M. Amin · Raviraj Thakur · Seth Madren · Han-Sheng Chuang · Mithuna Thottethodi · T. N. Vijaykumar · Steven T. Wereley · Stephen C. Jacobson

Received: 29 October 2012 / Accepted: 22 March 2013 / Published online: 10 April 2013

Abstract Current lab-on-a-chip (LoC) devices are assay-specific and are custom-built for each single experiment. Performing an experiment requires scientists or engineers to go through the time-consuming process of designing, fabricating, and testing a chip before conducting the actual experiment. This prolonged cycle can take months to complete, increasing effort and cost and reducing productivity. Similarly, minor modifications to an assay protocol re-incur the overheads of the design cycle. In this paper, we develop a *multi-purpose*, *software-programmableLab-on-a-Chip* (SPLoC), where the user simply writes or downloads a program for each experiment. We describe the components necessary to realize the SPLoC, which include a high-level programming language, an abstract instruction set, a runtime and control system, and a microfluidic device. We describe two key features of our high-level language compiler, and describe a novel variable-volume variable-ratio mixer. Finally, we demonstrate our SPLoC on four diverse, real-world assays.

A. M. Amin (✉)
Microfluidic Innovations, West Lafayette, IN 47906, USA
e-mail: ahmed@microfluidicinnovations.com

A. M. Amin · M. Thottethodi · T. N. Vijaykumar
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA

M. Thottethodi
e-mail: mithuna@purdue.edu

T. N. Vijaykumar
e-mail: vijay@ecn.purdue.edu

R. Thakur · S. T. Wereley
Birck Nanotechnology Center, School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA

S. T. Wereley
e-mail: wereley@purdue.edu

S. Madren · S. C. Jacobson
Department of Chemistry, Indiana University, Bloomington, IN 47405, USA

S. C. Jacobson
e-mail: jacobson@indiana.edu

H.-S. Chuang
Department of Biomedical Engineering, National Cheng Kung University, Tainan, Taiwan

1 Introduction

Lab-on-a-Chip (LoC) devices have been used in a plethora of applications, ranging from basic bio-chemistry research, to chemical synthesis, genomics, proteomics, clinical diagnostics and drug discovery. The requirement of smaller sample quantities, the increased accuracy and sensitivity of microfluidic operations, and the speed of performing once time-consuming protocols are some of the benefits realized by porting assays to microfluidic scale. Research on LoC devices can be broadly categorized into two main areas. First, the microfluidic research community has been actively engaged in developing and enhancing new processes and materials for the fabrication of LoCs, resulting in increased complexity and level of integration of chips. Multi-layered devices that integrate microfluidic valves and on-chip peristaltic pumps have been used for more complex assays. Similarly, the sophistication of operations that can be performed on-chip has evolved, from basic reservoirs and diffusion-based mixers, to chaotic mixers, complex fluid routing, and on-chip capillary electrophoresis. The integration of on-chip sensing capabilities, such as colorimetric and florescence detection, electrical sensing, and the use of antibodies immobilized on magnetic beads or gold nano-particle arrays have increased the range of applications that can now be performed at the microfluidic scale.

Second, the assay development and research community has been actively developing chips for new assays and improving chip design for existing assays. Although the end-result is typically a new protocol or modifications to known protocols, most of the effort in achieving this end goal is spent in the *design* of the LoC rather than the actual assay development. To test a new microfluidic-scale assay, scientists and engineers must identify the right microfluidic components to place on the chip, component parameters (e.g., channel width, mixer dimensions, etc.) and the layout of these components. Next, the scientist has to fabricate the chip using carefully selected fabrication processes, which typically require skilled expertise and expensive capital equipment. For more complex designs that require external control (such as microfluidic valves), the scientist has to develop a control platform, custom-written software and world-to-chip interfaces between the chip and external control equipment. Only then is the scientist able to run the assay and test the new protocol or validate a hypothesis. Any minor modifications to the assay or chip design require another design–fabricate–test cycle. This cycle can take anywhere from weeks to years. Moreover, the assay developer requires significant microfluidic expertise, intensive collaboration with a microfluidic expert, or contracting the chip design and manufacturing to expensive industrial third-parties.

The purpose of the work presented here is to attempt to bridge the gap between these two research areas in an abstract manner that reduces the *time, cost and expertise* required by users to develop new, microfluidic-scale assays, without having to worry about microfabrication details or electronic and software control. Though some approaches in the literature have attempted to improve one or more aspects of the design cycle, none provide a complete solution. For example, Su et al. (2006) have developed CAD tools to speed up the design of LoCs, which can then be sent to the fabrication service companies discussed above. Shaikh et al. (2005) have developed a breadboard-style kit where modular microfluidic components can be connected to create a LoC. However, assay design still takes on the order of days, and requires some manual labor for connecting the components together. Urbanski et al. (2006) have replaced these limited approaches with the pioneering idea of making LoC devices fully software-programmable. We extend their work to realize a software-programmable, continuous-flow multi-purpose lab-on-a-chip (SPLoC) platform. Our previous work has focused on defining the SPLoC hardware and the operations supported by the hardware that can be used by the software (Amin et al. 2007a, b) and key features of our compiler which translates assays written in our high-level language (HLL) to the low-level hardware operations (Amin et al. 2008). The SPLoC platform allows the user to program an assay in a few hours, rather than spend weeks and months to design, fabricate and test a microfluidic chip. The program is written in a simple, HLL, and is automatically compiled and executed on a highly integrated microfluidic chip. Minor modifications to assays are as simple as changing a few lines of code, rather than redesigning a chip from scratch. We note that there has been an increasing trend of similar work since (Fidalgo and Maerkl 2011; Jokerst et al. 2010). Given the numerous combination of all possible types of sensors, separators, detection properties, etc., a single universal device capable of running *all* possible assays is not feasible (Amin et al. 2007a). Rather, we envision several “classes” of chips, where each class can support a large subset of assays within one or two domains (e.g., genomics, proteomics).

The SPLoC platform consists of four main components. First, a HLL is used to define the assay. Similar to a computer programming language, the HLL is a structured collection of data types, variable definitions, and functional statements. For ease of use, we have carefully designed our language to resemble statements commonly used in the literature to define assay protocols. For example, mixing two fluids for 30 s is written as “MIX a:b IN RATIOS 1:2 for 30 s” in our HLL.

To allow the same HLL to be used across many classes of chips, the HLL is compiled into a standard intermediate language, called the AquaCore Instruction Set (AIS) (Amin et al. 2007b), which is the second component of our system. The AIS abstraction is analogous to assembly languages in computers (e.g., Intel’s x86 instruction set), and acts as a standard intermediate abstraction between HLLs and different SPLoC implementations. Each HLL program is compiled automatically into an AIS program comprising simple functional steps, or instructions, each of which is directly implemented in the SPLoC hardware (e.g., mix and heat).

Third, a control and runtime system is responsible for interpreting and translating each runtime instruction into electrical signals that control valve states, pumps, activate sensors and perform other fluidic functions.

Finally, the fourth component is a highly integrated SPLoC on which the assay runs. Our SPLoC, shown in Fig. 1, integrates microfluidic-scale *fluidic functional units* (*FFUs*) that are used by a large subset of assays: reservoirs, on-chip peristaltic pumps, an incubator serpentine, and sensor detection windows. All the components are connected by microfluidic channels and fluid control is achieved by microfluidic valves. As mentioned previously,

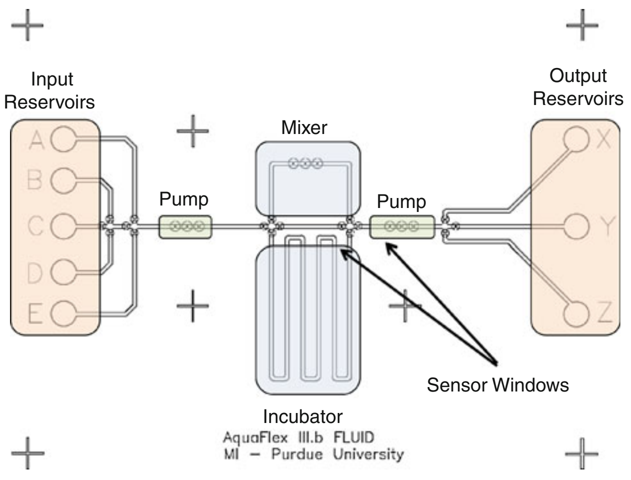


Fig. 1 Fluidic layout of SPLoC

by using the same abstraction layer, we can successfully execute different assays on the same chip implementation.

In addition to these standard components which have been demonstrated in the literature, we also integrate a variable-volume, programmable mixer. Mixing of two or more fluids is a key operation in almost all assays performed on LoCs. Typically, assay protocols dictate that fluids are mixed in certain ratios, and, in some cases, in certain absolute volumes. The microfluidic literature contains a plethora of mixing techniques each with their pros and cons (Nguyen and Wereley 2002; Handique and Burns 2001). However, because the mixers are designed for specific assays, such mixing techniques typically assume fixed ratio or fixed-volume mixing, and are designed as such. For a truly programmable LoC, the mixer must be capable of variable-ratio and variable-volume mixing. Additionally, the mixer should produce a homogeneous mixture in a relatively short period of time. We propose and demonstrate an active variable-volume, variable-ratio mixer that can meter and mix two or more fluids up to the volume of the mixer. Active mixing may introduce an air phase or bubbles which can significantly increase the time required for mixing—in some cases preventing mixing all together—or reduce the homogeneity of the resultant mixture. To address this issue, we propose a simple degassing method that is easy to fabricate and integrate, and eliminates the air phase problem, thus realizing an efficient and fully programmable variable-volume mixer.

To demonstrate the SPLoC concept, we have successfully fabricated several SPLoC prototypes, and programmed three diverse assays and run them on our SPLoC: (1) a glucose-level detection assay, (2) a basic enzyme kinetics assay, and (3) a particle sorting assay. Finally, we briefly describe bacteria culturing and automated synchronization work done by our collaborators.

2 Experimental

2.1 Microfluidic device fabrication

Three-layer devices were fabricated in polydimethylsiloxane (PDMS). We used standard soft lithography methods for the fluid and control channels (Duffy et al. 1998; Unger et al. 2000). Briefly, an SU8 mold was fabricated as follows. Glass wafers were cleaned and SU8-2025 (MicroChem Corp., Newton, MA) was spun at a speed of 4,000 rpm. The SU8 was pre-baked at 65 °C for 4 min and 95 °C for 7–8 min, followed by an exposure step at 225 mJ/cm^2. The wafer was post-baked at 95 °C then developed in SU8 developer for 100–120 s. Next, the wafer was hard baked at 225 °C for 35–40 min. To prepare the PDMS layers, the mold was placed in a petri dish and a 1:10 mixture of Sylgard 184 (Dow Corning Corp., Midland, MI) curing agent to pre-polymer was cast over the mold and baked in a convection oven at 70 °C for 3 h. The PDMS layers were carefully peeled off the mold, and had channel depths of 20 μm. A PDMS membrane was spun onto a clean glass wafer at 1,200 rpm and cured using the same conditions as the other layers. The membrane and control membrane were plasma treated then irreversibly bonded. Fluid and control holes were punched using a 22 gauge needle. Next, the fluid layer was carefully aligned on top of the assembly. Figure 1 shows the fluidic layer layout.

2.2 Runtime and control system

Figure 2 shows the electronic and pneumatic hardware control system (designed by Microfluidic Innovations, West Lafayette, IN, USA), and a manifold where microfluidic chips are inserted. The control system is composed of electronic boards, solenoid valves, pressure and vacuum sources (for actuating microfluidic valves), integrated voltage sources and off-chip sensors (e.g., thermocouple readers), and a microcontroller to interpret the runtime code and orchestrate the operation of the fluidic and other components. The manifold has ports that align with the chip control layer and thin tubes are used to connect the manifold to the controller.

2.3 Glucose level detection

To measure the glucose level in a sample, we use a fluorescence-based enzyme assay kit (k606-100 Glucose Assay Kit, BioVision). By mixing an enzyme with a glucose-containing sample, the glucose is oxidized and a by-product of the reaction reacts with a dye to generate color and fluorescence ($\lambda_{ex}/\lambda_{em} = 535/587$ nm). The generated color and fluorescence intensity were directly proportional to the glucose concentration in the sample, and were measured

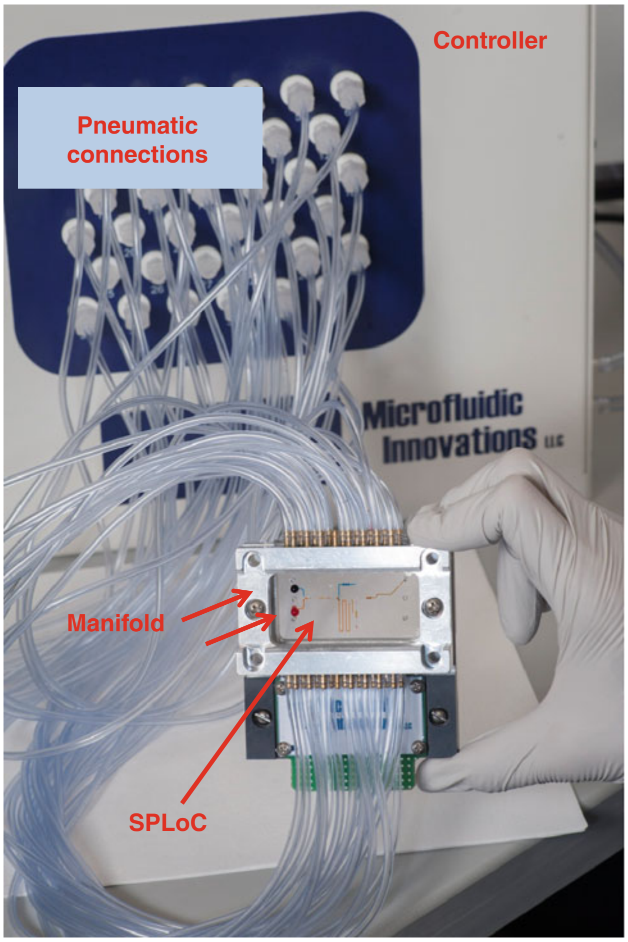


Fig. 2 Controller platform

with a microscope equipped with appropriate filters and a 12-bit CCD camera.

2.4 Enzyme kinetics

Histone deacetylase 2 (HDAC2) assays were performed with the Amplite Fluorimetric HDAC Activity Assay kit (AAT Bioquest, Sunnyvale, CA, USA) and purified HDAC2 enzyme (BPD Bioscience, San Diego, CA, USA). The substrate solution contained 2.8 µL of HDAC Green substrate and 13.8 µL of signal enhancer in 358.4 µL of assay buffer. Initially, the substrate solution, 10 µM Trichostatin A (TSA), 3 ng/mL HDAC2 enzyme and assay buffer were placed into different reservoirs on the microfluidic device. Measurements of the fluorescence intensity ($\lambda_{ex}/\lambda_{em} = 460/550$ nm) were conducted on a microscope and 16-bit CCD camera.

2.5 Particle sorting

Particle samples were composed of 0.5 µm particles and 3 µm particles at concentrations of 5.57×10^6 and 8.11×10^4/µL, respectively. Fluorescence intensities ($\lambda_{ex}/\lambda_{em} = 542/612$ nm) were measured using a microscope and 12-bit CCD camera.

2.6 Bacteria culturing and automated synchronization

Please refer to experimental details in Madren et al. (2012).

3 SPLoC platform

As we previously mentioned, the SPLoC platform consists of: (a) a high-level programming language and compiler; (b) an instruction set architecture that acts as an interface between the programming interface and microfluidic chip; (c) a control and runtime system, that interprets "programs" downloaded onto the platform, while supervising their *automatic* execution on the microfluidic chip; and (d) a software-programmable, multi-purpose microfluidic chip. In this section, we present a top-down detailed breakdown of these components.

3.1 High-level language and optimizing compiler

A programmable hardware device such as SPLoC without an accompanying compiler (that translates from high-level program to low-level hardware) is incomplete and falls well short of SPLoC's programmability goal and of the SPLoC's potential benefits. To that end, to allow users to write complex assays without implementation-specific knowledge, and to reduce the time and cost required for scientists to try new assays, we developed a high-level programming language (HLL). The HLL is analogous to a computer programming language such as C/C++ or Java. The HLL is defined as a set of declaration statements and a set of functional statements. The compiler was written with standard compiler techniques, from lexical analysis and semantic parsing to code generation, and shares many common procedures with conventional computer compilers (Cooper and Torczon 2004). For example, reservoir and FFU allocation for SPLoCs is similar to register allocation in conventional compilers. During the definition of the language, our priorities were (1) to simplify the language, to make it similar to how life scientists write assay protocols in lab procedures or scientific literature; and (2) to make the language as easy as possible to add new instructions or high-level constructs in the future to support new devices, sensors or operations. In addition to the fluid instructions, the compiler supports basic "dry-code" instructions that are used in conventional computer programs, such as arithmetic and logical expressions (+, −, *, /), and control flow constructs (if-then-else, while-loop). Such constructs enrich the language and allow more complex

(a)

```
ASSAY Glucose START
fluid Glucose, Reagent, Sample;
fluid a, b, c, d, e;
VARResult[5];
input Glucose 50;
input Reagent;
input Sample 30;
conflict Sample FOLLOWS Glucose WASH water;
a=MIX Glucose AND Reagent IN RATIOS1 : 1 FOR 10;
SENSE OPTICAL it INTOResult[1];
b=MIX Glucose AND Reagent IN RATIOS1 : 2 FOR 10;
SENSEOPTICAL it INTOResult[2];
c=MIX Glucose AND Reagent IN RATIOS1 : 4 FOR 10;
SENSEOPTICAL it INTOResult[3];
d=MIX Glucose AND Reagent IN RATIOS1 : 8 FOR 10;
SENSE OPTICAL it INTOResult[4];
e=MIX Sample AND Reagent INRATIOS1 : 1 FOR 10;
SENSE OPTICAL it INTOResult[5];
END
```

(b)

```
Glucose{
--input s3,ip3 ;Glucose;
--input s4,ip4 ;Reagent;
--input s5,ip5 ;Sample;
--washport w1     ;
move mixer1, s3,4.47;
move mixer1, s4,4.47;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(1);
move mixer1, s3,2.98;
move mixer1, s4,5.96;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(2);
move mixer1, s3,1.79;
move mixer1, s4,7.15;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(3);
move mixer1, s3,0.99;
move mixer1, s4,7.95;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(4);
wash w1;
move mixer1, s5,4.47
move mixer1, s4,4.47;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(5);
}
```

Fig. 3 Aqua source code and AIS compiled instructions. **a** Aqua code for a glucose calibration and detection assay, **b** compiled AquaCore instruction set code

assays to be written. For example, an assay may repeat several fluidic operations until a combination of values from several sensors satisfy a certain equation.

After the assay "program" is written by the scientist, it is compiled to produce intermediate, assembly-like code. A glucose-level detection assay written in the HLL and the compiler-generated AIS assembly code are shown in Fig. 3. The compiler is responsible for reading the HLL program, checking its correctness and parsing the program. During the compilation process, the compiler performs one or more passes. Whereas compilers for digital computers perform extra passes to optimize execution time and memory usage, our compiler targets problems specific to microfluidics—not found in digital computers—namely, variable-volume management and contamination mitigation. We briefly summarize these compiler passes, and refer the reader to Amin et al. (2007a, 2008) for computer-science-related details. The issue of fluid volume management arises because fluids have a fixed total volume, and the use of a fluid (variable) depletes it. If there are many uses of a fluid, the given volume of the fluid must be distributed carefully among the uses to prevent execution from running out of the fluid before all of the uses occur. This distribution poses a challenge when the uses require different proportions of volumes as is the case when a fluid is mixed with different fluids in different ratios (e.g., one use for a fluid is in a mix ratio of 1:2, whereas another use for the same fluid is in a mix ratio of 1:10). Dealing with fluid depletion is further complicated by low-level, implementation-dependent details of the fluidic hardware, such as maximum capacity (of reservoirs and functional units) and minimum fluid transport resolution (imposed by the fluid transport/handling hardware). Forcing the programmer to handle these constraints would diminish the practicality of SPLoCs. Consequently, our compiler passes handles this issue automatically, and assigns each fluid use a fixed volume that ensures that no underflow or overflow will occur.

The second issue of fluid contamination mitigation arises because fluid flow is not perfect and leaves residue in the SPLoC's components (e.g., channels and reservoirs) despite efforts to reduce such residue [e.g., reducing "dead spots" in reservoirs (Unger et al. 2000)]. This residue may contaminate the next fluid that uses the same components. Although an application-specific LoC can dedicate resources for the cross-contaminating reagents in the target assay, the SPLoC has no specific target assay or knowledge of specific cross-contaminating reagents and must be designed to share the same component among multiple fluids of the same assay. While the SPLoC programmer could manually insert wash steps in the assay to remove contamination, doing so would be cumbersome and would require the programmer to know the low-level details of which fluid uses which components and when. Instead, we require the programmer to specify only which fluid may contaminate which other, but not wash steps in precisely the correct places in the assay. The compiler automatically generates the required wash steps to avoid contamination. For example, line 8 in Fig. 3a specifies that if the sample follows glucose then there is a potential for contamination; and the compiler generates the required wash instructions in the AIS code in line 26 in Fig. 3b. Note that for some types of assays, the same SPLoC may be reused several times (with adequate washing and cleansing), and our proposed solutions may be extended for *inter*-assay contamination mitigation. For assays where chips may not be reused across assays, cheaper disposable SPLoCs may be used. However, that does not obviate the need of our contamination mitigation compiler pass which handles *intra*-assay contamination.

3.2 AquaCore instruction set (AIS)

To abstract the control details of the chip, and to support multiple "hardware" SPLoC classes, we propose the AquaCore Instruction Set (AIS) (Amin et al. 2007b). Figure 3b shows the AIS code for the assay in Fig. 3a. AIS serves a similar purpose as instruction sets in computers (e.g., Intel's x86 instruction set). By standardizing the instruction set at this abstraction level, other research and commercial groups may develop their own high-level languages, compilers and optimizers that compile their programs into AIS. Similarly, labs and industrial manufacturers designing complex chips may use the same AIS code to control their assays on these chips.

The next step in realizing practical SPLoCs is to automatically execute AIS instructions on microfluidic devices. To orchestrate the electrical control signals that actuate solenoid valves (which in turn control microfluidic valves and pumps) and to activate sensors, electrochemical detectors, incubators, or to read sensor values, a runtime system is required. The runtime system is, in its simplest form, analogous to the control unit of a microprocessor that executes microcode, acting as an interface between AIS code and physical hardware. To simplify the runtime system design and implementation, an AIS *assembler* translates each AIS instruction into one or more micro-sequence "runtime instructions". A runtime instruction represents a single state of valves, pumps or FFU configurations. For example, the AIS instruction "mov mixer1, s1, 10" which moves fluid from reservoir s1 to mixer1, is actually comprised of the following series of micro-instructions:

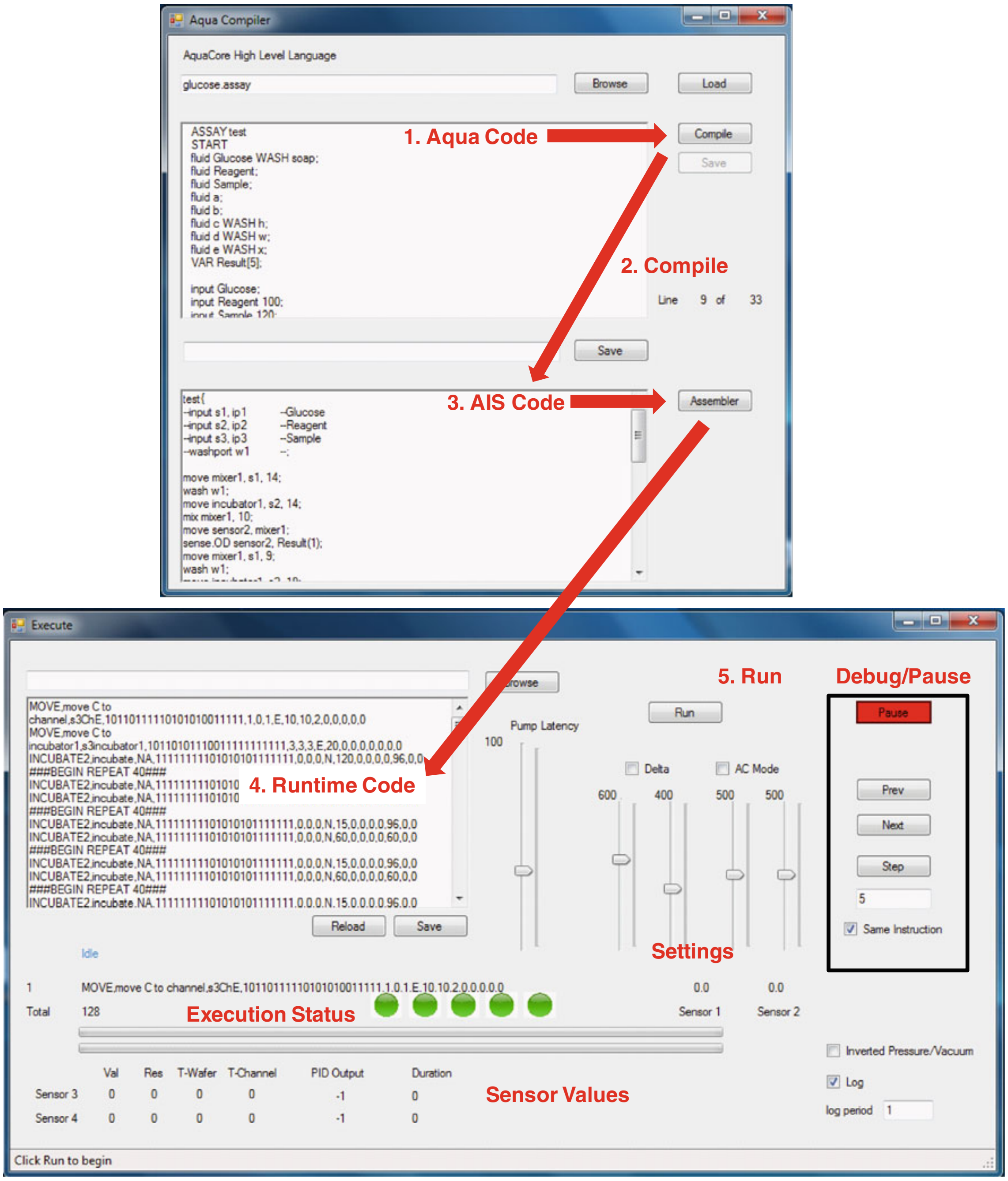
1. move the fluid from reservoir s1 to mixer1 input (position sensor)
2. meter 10 units of fluid into mixer1
3. return to s1 (or dispose to waste) any excess fluid in the channel.

The runtime system interprets each line and activates the corresponding relays, sensors and other electrical controls. Instructions complete when the number of pump actuations completes, or the sensor conditions are satisfied. Because different chips will differ in their implementation, to generate chip-specific runtime instructions (similar to a computer "binary" file), the assembler needs a backend *chip description file* (*CDF*). A CDF lists the number of reservoirs, FFU (e.g., mixers, incubators, sensors, channels and ports) and the connectivity between such units. For example, the configuration file would list the fluid path between reservoir s1 and mixer1 as a string of 1 and 0s that represent the valve states (open/closed) and the pump directions that would allow fluid to flow from s1 to mixer1. In contrast, the path between mixer1 and s1 would have the same valve state, but the pump directions would be reversed. Additionally, a CDF may list other implementation-specific details, such as maximum temperature support or sensor voltage ranges.

Figure 4 shows a graphical user interface (GUI) application that combines the full assay-writing and execution flow. First, the user writes an assay in the HLL, and the compiler compiles it to AIS assembly. The AIS code is then automatically assembled into runtime system microinstructions. Parameter files for the backend code generation and runtime system such as FFU connectivity maps, number of reservoirs, runtime parameters, etc., are read from text files. Finally, the resulting "binary" code can be loaded into the runtime system, and the control platform executes the assay on the SPLoC.

4 Microfluidic components

A multi-purpose microfluidic chip contains microfluidic units that can perform functions such as mixing, incubating (Chuang and Wereley 2009), and sensing (Nguyen and Wereley 2002; Hadd et al. 1997). These units are connected by one or more microfluidic channels and fluid flow is controlled with miniature microfluidic valves (Unger et al. 2000; Hasselbrink et al. 2002; Grover et al. 2003). Specifically, we use the valves designed by Grover et al. at Berkeley. Fluid movement is achieved by on-chip peristaltic pumps in each of the channels, where each pump comprises three valves. Chips consist of three layers: a fluidic layer, a flexible membrane layer and a gas control layer (the Berkeley valves use glass layers for fluid and control whereas we use PDMs layers). To open or close a valve, a vacuum (or pressure) source is applied to the gas control layer, causing a deflection of the flexible membrane out of (or into) the fluidic channel, holding it open (or sealing it shut). To supply gas or vacuum to individual valves (or valves in pumps) each control channel is connected to an external, off-chip solenoid valve. Solenoid valves are actuated by electrical signals generated by the controller and



```
ASSAY test
START
fluid Glucose WASH soap;
fluid Reagent;
fluid Sample;
fluid a;
fluid b;
fluid c WASH h;
fluid d WASH w;
fluid e WASH x;
VAR Result[5];

input Glucose;
input Reagent 100;
input Sample 120;
```

```
test{
--input s1, ip1        --Glucose
--input s2, ip2        --Reagent
--input s3, ip3        --Sample
--washport w1          --;

move mixer1, s1, 14;
wash w1;
move incubator1, s2, 14;
mix mixer1, 10;
move sensor2, mixer1;
sense.OD sensor2, Result(1);
move mixer1, s1, 9;
wash w1;
move incubator1, s2, 10;
```

```
MOVE,move C to
channel,s3ChE,1011011111010101010011111,1,0,1,E,10,10,2,0,0,0,0,0
MOVE,move C to
incubator1,s3incubator1,10110101110011111111111,3,3,3,E,20,0,0,0,0,0,0,0
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,120,0,0,0,0,96,0,0
###BEGIN REPEAT 40####
INCUBATE2,incubate,NA,1111111101010
INCUBATE2,incubate,NA,1111111101010
###BEGIN REPEAT 40####
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,15,0,0,0,0,96,0,0
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,60,0,0,0,0,60,0,0
###BEGIN REPEAT 40####
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,15,0,0,0,0,96,0,0
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,60,0,0,0,0,60,0,0
###BEGIN REPEAT 40####
INCUBATE2,incubate,NA,111111111010101011111111,0,0,0,N,15,0,0,0,0,96,0,0
```

```
1       MOVE,move C to channel,s3ChE,1011011111010101010011111,1,0,1,E,10,10,2,0,0,0,0,0    0.0        0.0
Total   128                                                                               Sensor 1   Sensor 2
```

	Val	Res	T-Wafer	T-Channel	PID Output	Duration
Sensor 3	0	0	0	0	-1	0
Sensor 4	0	0	0	0	-1	0

Fig. 4 Aqua compiler interface (*top*), runtime system interface (*bottom*)

runtime system. Similarly, heaters, sensors and other fluidic units may be controlled by electrical signals. Recall from Sect. 3 that the HLL, AIS abstraction and runtime system are applicable to any implementation. Next, we describe a specific SPLoC implementation, and describe in detail a critical component of our chips—a novel variable-volume mixer.

We have implemented and tested an all-PDMS chip that consists of a bottom fluidic channel layer, a flexible membrane (for valves) and a top control layer. This SPLoC integrates fluid reservoirs, microfluidic channels, microfluidic valves, on-chip peristaltic pumps, a variable-volume mixer, incubator region, and a sensor window as shown in Fig. 1. Although such components have been previously demonstrated and studied individually in LoCs, our novelty lies in the programmable control of these components and in the variable-volume mixer, which we describe next.

4.1 Variable-volume programmable mixers

A key operation in almost all assays performed on LoCs is the mixing of two or more fluids. Typically, assay protocols dictate that fluids are mixed in certain ratios or in certain absolute volumes. The microfluidic literature contains a plethora of mixing techniques, each with their pros and cons (Nguyen and Wereley 2002; Handique and Burns 2001; Burns et al. 1998). In general, two fluid streams are passed through a Y- or T-junction into a micro-channel where the fluids mix by diffusion. The geometries of the input channels dictate a fixed, single mix ratio, which has to be determined prior to executing the assay. Recently, Cooksey et al. (2009) achieved variable mix ratios by adjusting the pressure of the input fluids. However, their method cannot achieve variable mix volumes. Previous papers also propose to achieve arbitrary mix ratios, by applying a series of fixed-ratio 1:1 mix steps (Urbanski et al. 2006; Thies et al. 2007). For example, to achieve a mix of A:B in ratios 1:3, A and B are mixed in a 1:1 ratio, then the result is further mixed with B in a 1:1 ratio. Therefore, this fixed-volume manipulation constraint (a) is wasteful due to the discarding of half the resultant volume each step, and (b) requires several iterations to achieve variable mix ratios. Further, their technique and other mixing approaches do not allow an air phase to exist in the mixing channel between fluid slugs.

The problem of programmable variable-volume mixing amounts to addressing the following challenges. First, the fluids to be mixed need to be transported from their reservoirs and aligned at the mixer entrance, followed by a metering step that transports the metered volume into the mixer. Second, any intermediate air phases between the fluid slugs need to be eliminated to allow the slugs to mix. Similarly, air bubbles generated during mixing, either because of biological samples or because of active pumps, need to be removed. Third, the fluids must be mixed until a homogeneous resultant slug is achieved. Finally, the fluids must be completely purged from the mixer, to prevent waste and to reduce the effect of cross-contamination for future mix steps utilizing the same mixer. All the above steps must be fully automated without human intervention.

We use our microfluidic valves to control which fluids are transported to the mixer and the channel path they take and we use our on-chip peristaltic pumps to transport fluid. To ensure that the fluid being pumped is aligned at the mixer entrance under program control, there are two methods: (1) a closed-loop system which monitors the fluid location (e.g., using position sensors that sense fluid presence) and signal the controller to stop pumping, or (2) using open-loop control which ensures that the fluid is correctly aligned after a given duration of time. While more elegant, a closed-loop system requires integration of extra sensors on the chip, accurate post-fabrication calibration of the sensors, and a complex control system. Therefore, we focus on a novel open-loop method. To align the fluid at the mixer entrance, all air in the channel prior to the mixer entrance *valve* must be expelled. Therefore, a method that allows air to pass while retaining liquid is required. Hosokawa et al. propose using a hydrophobic micro-capillary vent (HMCV), which is an array of miniature hydrophobic channels that allow air to escape while fluids do not pass (Hosokawa et al. 1999). However, to withstand high-pressure heads (e.g., our in-channel peristaltic pumps can generate >10 kPa) without leakage requires very small widths of the micro-capillary vents (~3–4 μm for 10 kPa pressure head). The small width results in high-aspect ratio features and introduces microfabrication difficulties and reduces yield for large-scale production. Additionally, Urbanski et al. (2006) have developed a PDMS microfluidic latch for aligning fluid samples, where a partially deflected valve allows a mineral oil phase to pass through the hydrophobic PDMS surface while retaining the sample fluid due to surface tension, thereby aligning the fluid at the valve location. However, the microfluidic latch requires careful control of the valve deflection by adjusting the pressure used to deflect the valve membrane. While in principle this method could be used with an air phase instead of an oil phase, it would increase the control complexity and would require calibration for different samples.

Instead, we propose to use an *orthogonal vent* (Chuang et al. 2012). Recall that SPLoCs contain a flexible membrane sandwiched between the fluid and control layers for valving and pumping. If the membrane is gas-permeable (e.g., PDMS), air in front of the liquid interface can be removed by applying a pressure difference across the membrane. For aligning a liquid interface with respect to a valve location, the valve is closed and liquid is pumped using the peristaltic pump towards the valve. Thus, air in front of liquid gets compressed at a pressure which equals the peristaltic pump pressure. As the trapped air escapes through the membranes, the liquid front eventually advances towards the valve, leading to fluid alignment as shown in Fig. 5.

In order to calibrate the system for programmable control, the time required for fluid alignment requires characterization. Darcy's law combined with the continuity equation easily shows that the trapped bubble (between the

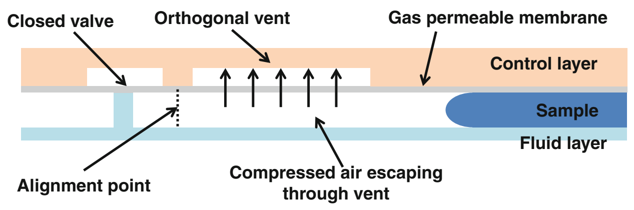


Fig. 5 Orthogonal vents

closed valve and liquid interface) diminishes following a first order exponential decay;

$$L(t) = L0 \times \exp(-\rho(P2 - P1)t)/(hd)$$

where $L(t)$ denotes the remaining trapped air length, $L0$ denotes the initial trapped air length, ρ stands for permeability of the membrane, $(P2 - P1)$ is the pressure difference across the membrane, h is the membrane thickness and d is the channel depth.

The time required to align the liquid slug at a desired location and expel the air in front of the slug can be reduced in four ways: (1) by increasing pressure difference across the membrane; (2) increasing permeability of the membrane; (3) reducing the membrane thickness; and (4) increasing the permeation area. Increasing the pressure difference may cause the bonding between the membrane and fluid layer to fail, resulting in fluid leakage. The permeability of the membrane was a parameter that we could not control without changing the material or modifying the surface characteristics. Because of practical limitations working with very thin membranes, we kept membrane thickness constant at 100 μm. Increasing the permeation area can be achieved by either (1) using wider channels, which causes significant deformations in the PDMS membrane for the same applied pressure/vacuum, or (2) using different geometries for the permeation area (Chuang et al. 2012). In summary, our prototype is able to expel an air slug of 15 mm in length and align the fluid from the reservoir with the valve in <25 s, using a 30 kPa vacuum applied to the orthogonal vents. Figure 6 shows the alignment and metering process on our device.

After the fluid is aligned at a known location, it can be metered into the mixer. Quake's group has proposed metering using valves (Unger et al. 2000). In their work, they fabricate valves that can be used for metering on the order of 100 pL with zero dead volume. Previous papers have also proposed using pump valves for metering (Grover et al. 2003; Chuang et al. 2008). As such, we also use the on-chip pumps to meter the required volume of fluid. The metering is achieved by the fact that each pump stroke displaces a certain fluid volume so that multiple strokes are used to achieve the required volume. Because the pumps

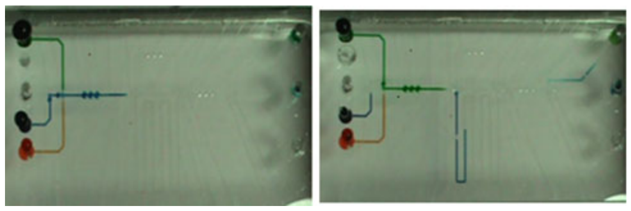


Fig. 6 Image showing alignment of blue fluid at entrance of mixer using the orthogonal vent (*left*). The alignment ensures that all of the metered *blue* fluid enters the mixer. The *blue* fluid has been metered, followed by alignment and metering of *green* fluid (*right*)

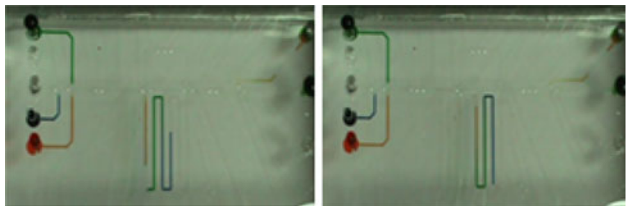


Fig. 7 Metered slugs with air trapped in between (*left*). Degassing air through orthogonal vents results in efficient mixing (*right*)

displace fairly consistent volumes, we did not see the need to make the metering any more accurate, as evidenced by the accuracy of our results for glucose concentration in the glucose assay (Sect. 5.1). We also note that fluid loss in channels during transport did not cause any significant deterioration of fluid volumes.

Automating the metering and mixing process, while ensuring that the entirety of the metered slug is in the mixer region prior to mixing without requiring user intervention, is not straightforward.

To meet this constraint, we align the fluid at the entrance of the mixer so that the fluid is metered as it *enters* the mixer, ensuring that the entirety of the metered slug is inside the mixer. For this reason, though metering can be done anywhere on the chip, it is done in SPLoC at the entrance of the mixer.

After two or more metered slugs are transported inside the mixer, the mixing step may begin. To reduce the time required for mixing, we use an active mixer that uses a peristaltic pump for mixing. Note that small air gaps may exist between the metered slugs, and, because our mixer supports variable volumes, an air phase will often exist inside the mixer. Further, because we use active mixers, small air bubbles may be generated during the mixing process. The existence of air bubbles between slugs will reduce the mixing efficiency and in some cases may prohibit mixing altogether. Trapped air bubbles in liquids will also reduce the homogeneity of the mixed product. Therefore, during mixing, the mixer must be *degassed* to remove any air bubbles between slugs. As such, we use the same orthogonal vent degassing method described earlier and apply a vacuum to the vents, to force air bubbles out of the fluid channel through the membrane and into the control channel as shown in Fig. 7. Thus, degassing occurs in multiple locations: wherever alignment is required (e.g., at the entrance of the mixer) and in the mixer.

The final step in mixing is to purge the mixer contents for use in another unit in the next instruction (e.g., transport to a sensor). Because the mixer may contain air, the location of the slug is unknown at the end of mixing. To align the slug, the exit valve is closed, and the mixer pumps are activated to push the slug toward the exit valve. Air between the slug and the exit valve will be expelled through the orthogonal vents, and the slug is purged from the mixer.

5 Assay development and testing

5.1 Glucose assay

This fluorescence-based assay establishes a calibration curve of fluorescence intensities for standard glucose concentrations, which can be used to determine the glucose concentration of unknown samples. Different glucose concentrations (0, 0.03, 0.06 and 0.09 nmol/μL) were prepared off-chip. A reaction mix consisting of enzyme, buffer and probe was also prepared. Each of the glucose concentrations and the reaction mix were placed into one of the SPLoC input ports. Figure 3 shows the software code. Upon execution, the reaction mix and each glucose concentration were mixed in a 1:2 ratio (approx. 200:400 nL), followed by an incubation step at room temperature. Finally, the resultant fluid was transported to a sensor location, which was placed under a microscope with a detector filter. At the sensing step, the fluorescence intensity was measured and recorded. To enhance the results, additional washing steps were inserted between fluid flows. The complete assay required approximately 2 h to complete with most of the time spent in the washing steps. Figure 8 shows the results of the sensing step for the different glucose concentrations. The fluorescence intensities are based on 512 grayscale levels. On average, the intensities for glucose concentrations of 0, 0.03, 0.06 and 0.09 nmol/μL were 135, 169, 229 and 314, respectively. For comparison, the average fluorescence intensities for background (i.e., no fluid) and the intensity of water (i.e., no glucose) were 128 and 130, respectively. The higher intensities recorded for the higher glucose-concentration samples indicate successful mixing, incubation, and assay execution.

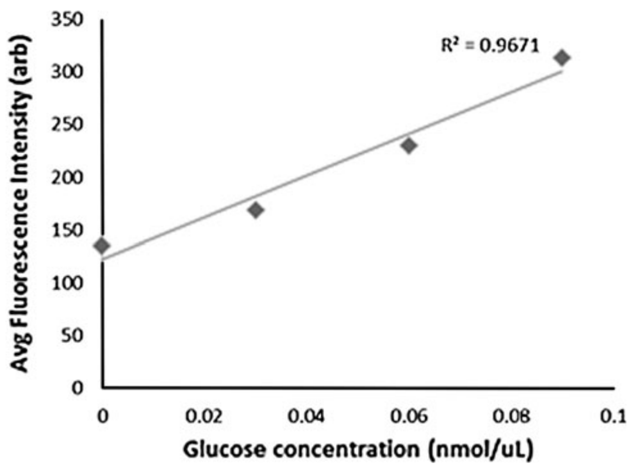


Fig. 8 Fluorescence intensity for a programmable automated glucose-level detection assay

5.2 Enzyme kinetics

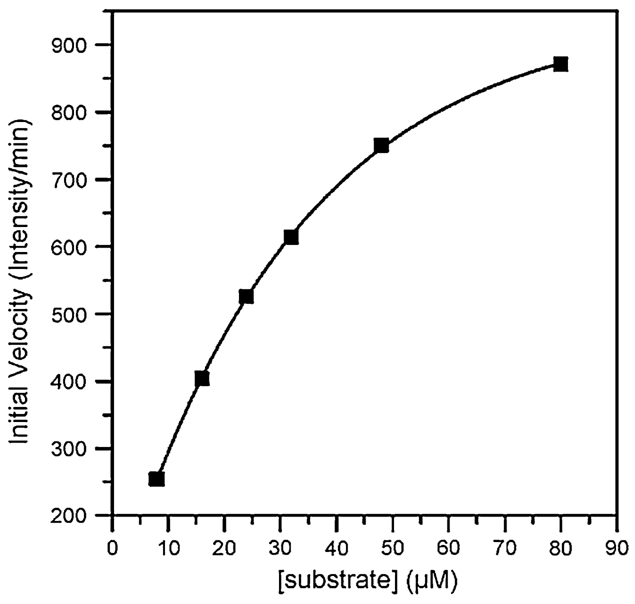
In a second assay, we examined the activity of purified histone deacetylase (HDAC2) enzyme with a real-time fluorescence assay. To test basic programmability, a program was written that performed the following operations. First, substrate was diluted by mixing with assay buffer. The ratio of dilution was varied in each run, resulting in substrate concentrations ranging from 8 to 80 μM. Next, the diluted substrate was mixed with a HDAC2 in a ratio of 1:1 for 3 min to achieve a final enzyme concentration of 1.5 ng/mL. The mixture was incubated and the fluorescence intensity was measured every 5 min for 25 min. All the above steps were automated and executed by the run-time system, inclusive of the alignment, metering and variable-volume mixing. Additionally, automatic contamination mitigation was performed by flushing out the main channel with air, then assay buffer, followed by air to empty the channel. Similarly, the incubator and mixer region were flushed between different sample runs. For high-level software code please refer to Amin (2011).

Figure 9 shows the saturation curve and Lineweaver–Burk plot for the HDAC2 enzyme. The K_m value derived from the Lineweaver–Burk plot is 28 μM, which is in agreement with the literature value of 32 ± 4 μM (Schultz et al. 2004). This assay confirms reproducible mixing, metering and dilution over an order of magnitude of concentration as well as no cross-contamination between runs.

To illustrate the ease with which more complex assays could be developed with the platform, the inhibition of HDAC2 with TSA was studied. The program first mixed substrate, TSA and assay buffer in the incubator region. Next, HDAC2 enzyme was metered into the mixer and mixed for 3 min. The concentrations of substrate and enzyme were 48 μM and 1.5 ng/mL, respectively, for all runs, while the TSA concentrations were varied in each run. As before, the mixture was incubated at room temperature and the fluorescence intensity was measured over 25 min at 5 min intervals. Figure 10 shows the inhibition of HDAC2 activity by the TSA.

5.3 Particle sorting assay

H-filters are an established and well-known method for particle sorting (Brody and Yager 1997). An input stream with different sized particles is flowed into the same channel with an extraction buffer. Smaller particles in the input stream diffuse more rapidly into the extraction stream, while larger particles take longer time to diffuse. Hence, at the end of the channel, the extraction stream contains some of the smaller particles that were originally present in the input stream. To enhance the efficiency of the separation, the remainder of the input stream is re-flowed again with the



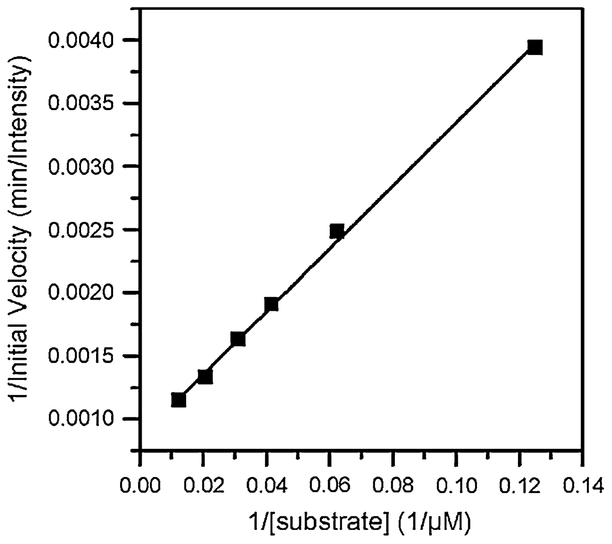


Fig. 9 Saturation curve for HDAC2 (*top*) collected with the automated PLoC program. The initial velocity was calculated with the fluorescence intensity collected 15 min after the mixing. Lineweaver–Burk plot (*Bottom*) used to determine a K_m value of 28 μM. This is in agreement with previously determined values

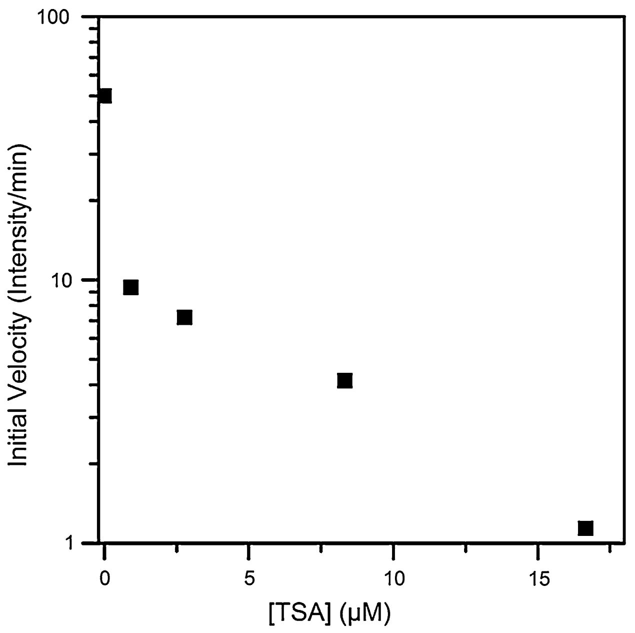


Fig. 10 Inhibition of HDAC2 enzyme activity by TSA. The initial velocity was calculated with the fluorescence intensity collected 15 min after mixing

extraction buffer to allow more of the smaller particles to diffuse. By repeating this process multiple times, the original input stream now has a higher concentration of large particles, and the extraction stream has a higher concentration of smaller particles. To achieve this higher efficiency, other researchers have "cascaded" the H-filters (Kapishnikov et al. 2006). In contrast, given the programmable interface and SPLoCs, we presented a novel method in which the extraction is performed "recursively", where the same chip is used to automatically perform multiple cycles of extraction (Chuang et al. 2010). An added advantage of the programming interface is that adding wash steps between the extraction cycles to flush the channels and prevent cross-contamination (e.g., larger particles entering the extraction stream) was very simple (and could be automated). For software code please refer to Amin (2011), and for more details on the assay we refer the reader to our previous study (Chuang et al. 2010) and briefly summarize the results below.

The initial input stream comprised two main particle sizes: 0.5 μm particles and 3 μm particles at concentrations of 5.57×10^6 and 8.11×10^4/μL, respectively. The diffusion-based sorting to separate the different particle sizes—with smaller particles diffusing into the extraction stream—was repeated for 10 cycles. Figure 11 shows the original suspension and the extraction stream after two cycles and after ten cycles. Note that the concentration of small particles in the extraction stream is significantly higher after ten cycles. Some contamination (due to diffusion of larger particles) is also present in the extraction stream. The cumulative concentration versus cycle plot is shown in Fig. 12, expressed as a percentage relative to the original value. As expected, the concentration of small particles in the extraction stream increases as more cycles are executed, and goes from 7.5 % after two cycles to 38 % after 10 cycles.

5.4 Bacteria culturing and automated synchronization

In another study, our software and platform was used to culture, synchronize, and analyze bacteria (Madren et al. 2012). Cell-based assays that examine events that occur at different stages of the cell cycle require robust cell synchronization methods. For bacteria synchronization,

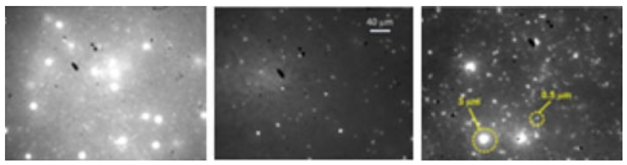


Fig. 11 Relative cumulative concentration of small (0.5 µm) and large (3 µm) particles at different sorting cycles

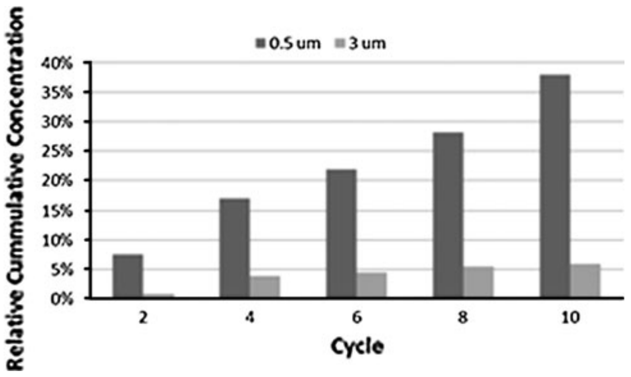


Fig. 12 Relative cumulative concentration of small (0.5 µm) and large (3 µm) particles at different sorting cycles

existing approaches require large volumes (>100 mL), are labor intensive, require long incubation times, result in inefficient collection of cells of the same age, or are limited to single collections. The SPLoC controller and software and a simple variant of SPLoC helped to overcome these limitations. This assay collects swarmer cells from a biofilm cultured on-chip at programmed intervals over a period of 3 days, resulting in populations that contain >80 % of swarmer cells, no human intervention, and significantly less media used. To illustrate the benefits of the SPLoC system, continuous operation of their device for 3 days would require 3 mL of media and could produce 432 synchronized cell populations with an automated program, whereas the existing plate-release technique, if it could even produce the same number of cell populations would require 43 L of media (Madren et al. 2012).

6 Conclusion and future directions

Analogous to programmable computers, the SPLoC concept offers the following significant benefits: (1) cost, design effort, and turn-around time reduction: the SPLoC eliminates the per-assay LoC design effort and turn-around time and thus reduces cost. The time to develop a new assay is reduced to specifying the assay and compiling it for the SPLoC. (2) Productivity: the benefits of the instruction set abstraction enables assay developers and SPLoC designers to concentrate on their area of expertise without requiring cross-disciplinary expertise and allows fabrication- and test-free prototyping of new assays. (3) Composition: complex assays can be constructed by invoking other assays like procedures or subroutines. This ability to compose larger assays using assay modules offers the benefit of modular design. Further, conditional invocation of assays depending on the outcome of previous steps can eliminate the human-in-the-loop, which minimizes the potential for human error and vastly improves throughput.

There is a lot of work in progress. First, we are working on modifying the incubator structure to reduce sample evaporation at high temperatures, with the goal of extracting amplified products after PCR. Second, we are working on integrating electrochemical detection capabilities with capillary electrophoresis channels, allowing analyte detection and quantification. Finally, we plan on extending our programming language and runtime system to run immunoassays that use on-chip magnetic beads.

Acknowledgments This work is based in part upon work supported by the National Science Foundation (NSF) under Grant Numbers CCF-0726821 and CCF-0726694. This work was also supported in part by the Purdue Research Foundation (PRF). Work at Microfluidic Innovations was supported by NSF Award IIP-1013625 and NIH grant 1R43EB012392-01. The authors would also like to thank the staff of the Birck Nanotechnology Center.

References

Amin AM (2011) A Programmable architecture and compiler for microfluidics. Doctoral Dissertation, Purdue University, West Lafayette

Amin AM, Thottethodi M, Vijaykumar TN, Wereley ST, Jacobson SC (2007a) AquaCore: A programmable architecture for microfluidics. In: Proceedings of the 34th international symposium on computer architecture (ISCA-2007), June 2007 pp 254–265

Amin AM, Thottethodi M, Vijaykumar TN, Wereley ST, Jacobson SC (2007b) Aquacore: A general-purpose architecture for programmable microfluidics. In: Proceedings of the 11th international conference on miniaturized systems for chemistry and life sciences (µTAS 2007), October 2007

Amin AM, Thottethodi M, Vijaykumar TN, Wereley ST, Jacobson SC (2008) Automatic volume management for programmable microfluidics. In: Proceedings of the ACM SIGPLAN conference on programming language design and implementation (PLDI-2008), June 2008 pp 56–67

Brody JP, Yager P (1997) Diffusion-based extraction in a microfabricated device. Sens Actuat A 58(1):13–18. doi:10.1016/s0924-4247(97)80219-1

Burns MA, Johnson BN, Brahmasandra SN, Handique K, Webster JR, Krishnan M, Sammarco TS, Man PM, Jones D, Heldsinger D, Mastrangelo CH, Burke DT (1998) An integrated nanoliter DNA analysis device. Science 282(5388):484–487. doi:10.1126/science.282.5388.484

Chuang HS, Wereley ST (2009) Design, fabrication and characterization of a conducting PDMS for microheaters and temperature sensors. J Micromech Microeng 19(4):045010. doi:10.1088/0960-1317/19/4/045010

Chuang HS, Amin AM, Wereley ST, Thottethodi M, Vijaykumar TN, Jacobson SC (2008) Polydimethylsiloxane (PDMS) peristaltic pump characterization for programmable lab-on-a-chip applications. In: Proceedings of the 12th international conference on miniaturized systems for chemistry and life sciences 12

Chuang HS, Jacobson S, Wereley ST (2010) A diffusion-based cyclic particle extractor. Microfluid Nanofluid 9(4):743–753. doi:10.1007/s10404-010-0589-0

Chuang HS, Raviraj T, Wereley ST (2012) Characterizations of gas purge valves for liquid alignment and gas removal in a microfluidic chip. J Micromech Microeng 22(8):085023. doi:10.1088/0960-1317/22/8/085023

Cooksey GA, Sip CG, Folch A (2009) A multi-purpose microfluidic perfusion system with combinatorial choice of inputs, mixtures, gradient patterns, and flow rates. Lab Chip 9(3):417–426. doi:10.1039/B806803H

Cooper K, Torczon L (2004) Engineering a compiler. Morgan Kaufmann, San Francisco

Duffy DC, McDonald JC, Schueller OJA, Whitesides GM (1998) Rapid prototyping of microfluidic systems in poly(dimethylsiloxane). Anal Chem 70(23):4974–4984. doi:10.1021/ac980656z

Fidalgo LM, Maerkl SJ (2011) A software-programmable microfluidic device for automated biology. Lab Chip 11(9):1612–1619. doi:10.1039/C0LC00537A

Grover WH, Skelley AM, Liu CN, Lagally ET, Mathies RA (2003) Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices. Sens Actuator B-Chem 89(3):315–323. doi:10.1016/S0925-4005(02)00468-9

Hadd AG, Raymond DE, Halliwell JW, Jacobson SC, Ramsey JM (1997) Microchip device for performing enzyme assays. Anal Chem 69(17):3407–3412. doi:10.1021/ac970192p

Handique K, Burns MA (2001) Mathematical modeling of drop mixing in a slit-type microchannel. J Micromech Microeng 11(5):548. doi:10.1088/0960-1317/11/5/316

Hasselbrink EF, Shepodd TJ, Rehm JE (2002) High-pressure microfluidic control in lab-on-a-chip devices using mobile polymer monoliths. Anal Chem 74(19):4913–4918. doi:10.1021/ac025761u

Hosokawa K, Fujii T, Endo I (1999) Handling of picoliter liquid samples in a poly(dimethylsiloxane)-based microfluidic device. Anal Chem 71(20):4781–4785. doi:10.1021/ac990571d

Jokerst JV, Jacobson JW, Bhagwandin BD, Floriano PN, Christodoulides N, McDevitt JT (2010) Programmable nano-bio-chip sensors: analytical meets clinical. Anal Chem 82(5):1571–1579. doi:10.1021/ac901743u

Kapishnikov S, Kantsler V, Steinberg V (2006) Continuous particle size separation and size sorting using ultrasound in a microchannel. J Stat Mech. doi:10.1088/1742-5468/2006/01/P01012

Madren SM, Hoffman MD, Brown PJB, Kysela DT, Brun YV, Jacobson SC (2012) Microfluidic device for automated synchronization of bacterial cells. Anal Chem 84(20):8571–8578. doi:10.1021/ac301565g

Nguyen N, Wereley ST (2002) Fundamentals and applications of microfluidics. Artech House, Norwood

Schultz BE, Misialek S, Wu J, Tang J, Conn MT, Tahilramani R, Wong L (2004) Kinetics and comparative reactivity of human class I and class IIb histone deacetylases. Biochem 43(34):11083–11091. doi:10.1021/bi0494471

Shaikh KA, Ryu KS, Goluch ED, Nam J-M, Liu J, Thaxton CS, Chiesl TN, Barron AE, Lu Y, Mirkin CA, Liu C (2005) A modular microfluidic architecture for integrated biochemical analysis. PNAS 102(28):9745–9750. doi:10.1073/pnas.0504082102

Su F, Chakrabarty K, Fair RB (2006) Microfluidics-based biochips: technology issues, implementation platforms, and design-automation challenges. IEEE transactions on computer-aided design of integrated circuits and systems, 25 (2) pp 211–223. doi:10.1109/TCAD.2005.855956

Thies W, Urban JP, Thorsen T, Amarasinghe S (2007) Abstraction layers for scalable microfluidic biocomputing. Nat Comp. doi:10.1007/s11047-006-9032-6

Unger MA, Chou HP, Thorsen T, Scherer A, Quake SR (2000) Monolithic microfabricated valves and pumps by multilayer soft lithography. Science 288(5463):113–116. doi:10.1126/science.288.5463.113

Urbanski JP, Thies W, Rhodes C, Amarasinghe S, Thorsen T (2006) Digital microfluidics using soft lithography. Lab Chip 6(1):96–104. doi:10.1039/B510127A