Fluigi: Microfluidic Device Synthesis for Synthetic Biology

HAIYAO HUANG and DOUGLAS DENSMORE, Boston University

One goal of synthetic biology is to design and build genetic circuits in living cells for a range of applications. Our incomplete knowledge of the effects of metabolic load and biological "crosstalk" on the host cell make it difficult to construct multilevel genetic logic circuits in a single cell, limiting the scalability of engineered biological systems. Microfluidic technologies provide reliable and scalable construction of synthetic biological systems by allowing compartmentalization of cells encoding simple genetic circuits and the spatiotemporal control of communication among these cells. This control is achieved via valves on the microfluidics chip which restrict fluid flow when activated. We describe a Computer Aided Design (CAD) framework called "Fluigi" for optimizing the layout of genetic circuits on a microfluidic chip, generating the control sequence of the associated signaling fluid valves, and simulating the behavior of the configured biological circuits. We demonstrate the capabilities of Fluigi on a set of Boolean algebraic benchmark circuits found in both synthetic biology and electrical engineering and a set of assay-based benchmark circuits. The integration of microfluidics and synthetic biology has the capability to increase the scale of engineered biological systems for applications in DNA assembly, biosensors, and screening assays for novel orthogonal genetic parts.

Categories and Subject Descriptors: J.6 [**Computer-Aided Engineering**]

General Terms: Design

Additional Key Words and Phrases: Synthetic biology, microfluidics, genetic circuits

ACM Reference Format:
Haiyao Huang and Douglas Densmore. 2014. Fluigi: Microfluidic device synthesis for synthetic biology. ACM J. Emerg. Technol. Comput. Syst. 11, 3, Article 26 (December 2014), 19 pages.
DOI: http://dx.doi.org/10.1145/2660773

1. INTRODUCTION

Synthetic biology seeks to harness and manipulate existing natural systems for novel applications using to engineering principles. The primary paradigm in the field is to identify biological primitives involved in the transformation of DNA to a protein and encapsulate these individual DNA sequences as *parts* [Voigt 2006]. Synthetic biology has taken advantage of the modularity in the structure of bacterial open reading frames (ORFs) to abstract portions of the DNA sequence as *parts*, which include sequences for promoters, ribosome binding sites, coding regions, and transcriptional terminators. These parts can be combined to create functional *devices* which can be introduced into living organisms such as bacteria, yeast, or mammalian cells [Canton et al. 2008; Khalil et al. 2012].

1.1. Biological Logic Design

Since logical frameworks such as repressible systems exhibiting the behavior of Boolean inverters already exist in gene expression systems, these frameworks may

This work is supported by the Clare Booth Luce Graduate Fellowship.
Authors' address: H. Huang and D. Densmore, Electrical and Computer Engineering, Boston University, 8 St. Mary's St, Boston, MA, 02215; email: {huangh, dougd}@bu.edu.

DOI: http://dx.doi.org/10.1145/2660773

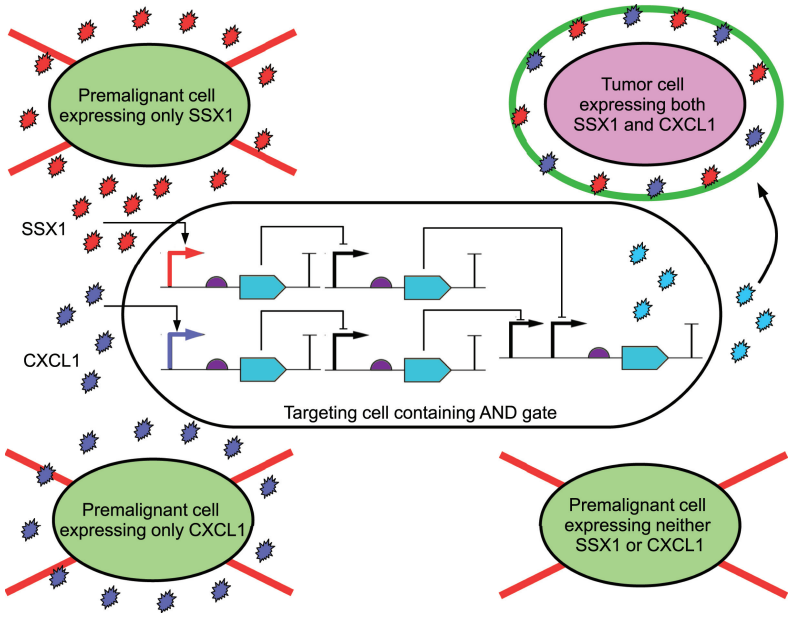


Fig. 1. Two proteins highly expressed in tumor cells but not premalignant cells are the synovial sarcoma X-breakpoint protein-1 (SSX1) and the inflammatory chemokine (CXCL1) [Nissim and Bar-Ziv 2010]. A biological circuit implementing a Boolean AND gate was used to differentiate malignant and nonmalignant cells based on their protein expression by only targeting cells that express both SSX1 and CXCL1.

be repurposed to detect specific combinations of chemical or environmental triggers for targeted pharmaceutical and biotechnology applications [Tamsir et al. 2011]. A summary of the various classes of genetic circuits for implementing Boolean logic functions in biology and their respective advantages and challenges is found elsewhere [Brophy and Voigt 2014]. We use the digital abstraction not to replicate silicon based computing, but as an alternative method for designing robust biological circuits that are insensitive to noise and can be tuned to specific input conditions. The digital abstraction is well understood, and numerous techniques have been developed for its description [De Micheli 1994], synthesis [Weste and Harris 2011], and verification [Hassoun and Sasao 2002].

One application for biological logic circuits is in the field of cancer research, where the use of digital logic provides the necessary specificity for targeting strains of cancer cells while leaving other cells unharmed [Ruder et al. 2011; Shankar and Pillai 2011]. We show in Figure 1 a genetic AND gate in the cancer detection system described in Nissim and Bar-Ziv [2010]. This system uses two transcriptional start sites that are highly active in only cancer cells to detect the differences between tumor cells and premalignant cells. These two sites were chosen from the following: the chromatin structural protein histone-H2A1 promoter, the synovial sarcoma X-breakpoint protein-1 SSX1 promoter, and the inflammatory chemokine CXCL1 promoter. A targeting cell containing a logic circuit would be able to determine malignancy and deliver a payload of drugs or otherwise induce cell death only to the tumor cells.

While devices implementing two input Boolean logic functions are useful in synthetic biology, more complex computation such as those involving more inputs would allow for applications such as the differentiation of molecular species, the identification of specific genetic markers, and environmentally tailored drug dosage responses [Purnick

and Weiss 2009]. More complex functions may be constructed by increasing the levels of logic in the genetic device. A four input transcriptional AND gate [Moon et al. 2012] with 11 orthogonal (noninterfering) regulatory proteins was constructed with this method from two levels of two input AND gates. However, introducing complex systems such as this into a cell presents numerous challenges.

1.2. Synthetic Biology Design Challenges

A fundamental problem in building more complex biological systems lies in the lack of orthogonal parts and signaling molecules available [Moon et al. 2012] and the difficulty of engineering new orthogonal parts. Using non-orthogonal parts in the same system will result in unwanted molecular interactions in the cell which may interfere with the intended function of the system. We refer to these unwanted interactions as *biological crosstalk*. As the number of unique regulatory proteins in the system increases, the number of factors to consider in the analysis and debugging of the system grows exponentially [Lu 2010]. In addition, the production of these regulatory proteins may also adversely affect cell metabolism [Klumpp et al. 2009].

These issues may be mitigated by distributing the computational load across many cells and separating large circuits into smaller circuits, each in a different cell colony, that communicate with each other through intercellular signaling chemicals [Tamsir et al. 2011; Regot et al. 2011; Macía et al. 2012]. Tamsir et al. [2011] built all possible two input Boolean functions from biological NOR gates in this manner. The colonies are spatially separated on a plate by hand, with the intercellular signaling chemicals spreading through diffusion. However, the diffusion of these signals is not directed towards specific colonies and may reach unintended targets and cause crosstalk between circuits. Increasing the number of intercellular signals to use in these systems does not fully address the problem of scalability as there is a lack of orthogonal intercellular signaling systems. We expand on the idea of distributed biological computing by using microfluidics to physically isolate each colony and restrict intercellular signaling to specific colonies via controlling the media flow to reduce this crosstalk.

1.3. Microfluidics Devices for Synthetic Biology

Microfluidics consists of the systems and tools for studying and manipulating small volumes of liquids. The decrease to microliter or smaller scales brings advantages such as allowing for more predictable fluid flow and decreasing the amount of reagents needed for reactions, as well as smaller test devices and experimental setups. The field of microfluidics covers a wide range of technologies such as lateral flow tests, linear actuated devices, pressure driven laminar flow, microfluidic large-scale integration, segmented flow microfluidics, centrifugal microfluidics, electrokinetics, electrowetting, surface acoustic waves, and dedicated systems for massively parallel analysis [Mark et al. 2010]. Due to the potential in microfluidic systems for the precise control over input stimuli [Wang et al. 2012; Dertinger et al. 2001], and the ability to track single cells [Ferry et al. 2011], there has been an increased interest in microfluidic platforms to solve current challenges synthetic biology [Lin and Levchenko 2012; Huang and Densmore 2014].

We have two primary criteria for our hypothetical microfluidic based distributed biological computing platform: the ability to physically compartmentalize growing cells and the ability to control and route chemical signals between cell colonies in those compartments. Secondary criteria include ease of device fabrication, ease of system testing, and ability for design automation. We considered several possible technologies, including silicon-based pressure-driven laminar flow systems and digital microfluidics before deciding to tailor our approach to use polydimethylsiloxane (PDMS)-based

pressure-driven laminar flow systems and microfluidic large-scale integration (mLSI). Silicon-based systems do not offer the signal control that mLSI can, and while digital microfluidics meet the criteria for cell culture and signal control, PDMS-based devices are easier to fabricate.

The properties of PDMS make it well suited for use in biological applications as it is optically transparent, chemically inert, impermeable to water while being permeable to gases, and nontoxic to cells [Sia and Whitesides 2003]. PDMS-based microfluidics have been used for a variety of purposes in recent years, including as an alternative platform for computation [Gehani and Reif 1999; Livstone et al. 2006; Thies et al. 2008]. Development of the microchemostat [Balagaddé et al. 2005] has allowed for cells to be grown in microfluidic chips for long periods of time, thus allowing for more complex, long term experiments.

Microfluidic chips are fabricated from PDMS through soft lithography [Duffy et al. 1998; Sia and Whitesides 2003]. Multilayer soft lithography extends this process to create devices from multiple layers of bonded PDMS [Unger et al. 2000]. A typical multilayer device consists of a flow layer and a control layer. Channels on the flow layer carry signals of interest while channels in the control layer are pressurized with external actuators. Valves in the control layer are placed over channels in the flow layer to control fluid flow. When pressurized, these valves deform the PDMS and create a seal across the fluid channel to restrict fluid flow [Amin et al. 2009]. This technology has already proven to be effective in isolating cell colonies and controlling intercellular signals between them [Liu et al. 2010].

1.4. A Combined Approach for Synthetic Biology and Microfluidics

The first step in creating a microfluidic platform for distributed biological computation is to build the necessary framework and tools to allow for easy iteration and refinement of our microfluidic design while taking into account the biological components. While rudimentary CAD tools for both microfluidic chips based on multilayer soft lithography [Amin et al. 2009; Minhass et al. 2012, 2013] and synthetic biology circuit design [Xia et al. 2011; Cai et al. 2010; Chandran et al. 2009] exist, there are currently no CAD tools that combine, formalize, and optimize the two processes. This article presents a microfluidic design framework which allows for simple cellular synthetic biological systems to communicate in a controlled fashion while keeping the cell types physically separated. By abstracting the biological details (which can be found in Endy [2005] and Andrianantoandro et al. [2006]) we focus primarily on basic Boolean algebraic biological computing elements and an intercellular signaling fluid. Assuming that individual genetic circuits representing Boolean algebraic functions can be put into cellular systems [Moon et al. 2012; Tamsir et al. 2011; Daniel et al. 2013; Brophy and Voigt 2014] and that intercellular communication mechanisms can be combined with these systems [Bonnet et al. 2012; Basu et al. 2005] challenges still remain in using these systems on a large scale. A summary of some of the challenges and potential solutions are shown in Figure 2.

A strength of this approach is that we can use existing paradigms in the electronic design automation community and apply them to microfluidics. Our results show that large sets of Boolean algebraic functionality can be realized with an automated software workflow using primitive genetic gates organized on a generic microfluidic valve based fabric. These systems should prove effective at removing the described crosstalk issues, which will allow for more formally engineered systems. This will also allow us to leverage existing CAD infrastructure and tools. This article focuses on describing how a design synthesis workflow, starting from a high-level description of desired functionality, can be used to place the biological components in a microfluidic system, route the signaling fluid, control the valve network, and simulate the expected control system behavior.

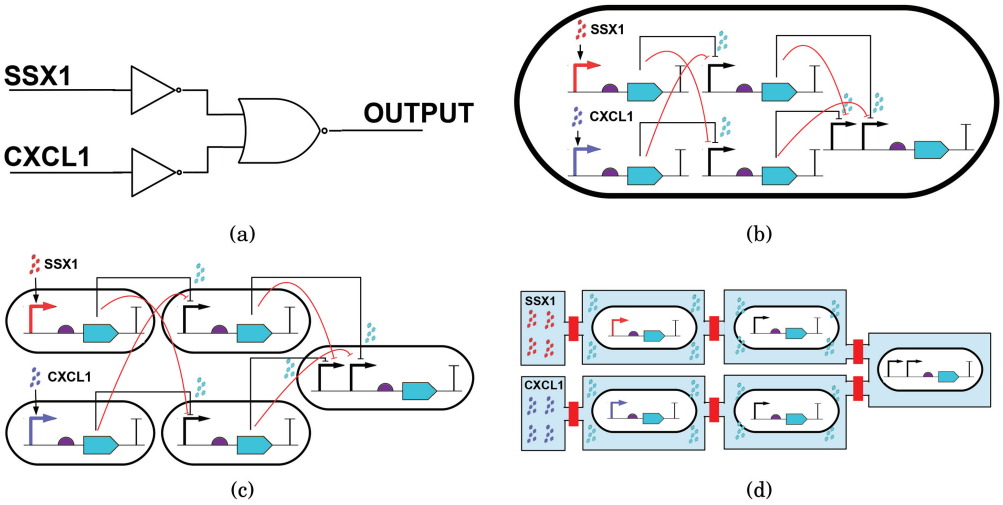


Fig. 2. Figure 2(a) shows an AND circuit made of NOT and NOR gates and Figure 2(b) shows the same function implemented by genetic circuits in a single cell. Bent arrows represent transcriptional start sites, and blue arrowed boxes represent genes. Black arcs indicate repression of the transcriptional start sites by the proteins produced while red arcs indicate potential crosstalk sites between the proteins. Separating the circuits into multiple cells will lessen the metabolic impact of the circuit on the host cell, but only shifts the problems arising from crosstalk to the intercellular signals (2(c)). The use of microfluidic chambers and valves to physically separate the genetic circuits (2(d)) eliminates crosstalk and allows the precise application of external control molecules.

2. DEFINITIONS

We define a GATE to be a group of cells that perform a specific function and have some number n of inputs and one output. For the examples shown in this article, $n = 1$ or $n = 2$. The top-level function we are trying to implement can then be defined as GATEs connected in a directed graph.

We define a CHIP as having two elements, a flow layer and a control layer. The flow layer represents the paths fluids can travel on the chip, and the control layer represents the valves used to manipulate fluid flow and the connections between them. The flow layer can be represented as a directed graph and the control layer as an undirected graph. We assume both these graphs are sparsely connected. A hypothetical flow layer graph and control layer graph for a sample chip is shown in Figure 4(a).

For the flow layer, we define a vertex in the graph to be NODE and an edge in the graph to be a CHANNEL. A CHANNEL represents a segment along which air, fluid, or cells may travel. A subset of channels are CELL TRAPs, where GATEs may be placed in the chip. A NODE has one or more channels entering or exiting it, and must be connected to at least one channel. A subset of NODEs are FLOW PORTs, which represent locations where the chip interfaces with the outside world. Flow ports may be sources, which have no incoming channels or sinks, which have no outgoing channels. Fluids enter the chip through source ports and exit the chip through sink ports.

For the control layer, we define a vertex in the graph to be a VALVE and an edge in the graph to also be a CHANNEL. A VALVE is a location on the control layer that intersects a channel in the flow layer with the intention of manipulating flow in the channel. Each valve is associated with a channel in the flow layer. A subset of valves are the CONTROL PORTs, which are locations where air enters and exits the chip. Unlike flow ports, air may enter and exit the same control port. Each valve must be connected to a control

Table I. Definition of Terms

Term	Symbol	Definition
CHIP	H	{FLOW LAYER, CONTROL LAYER}
FLOW LAYER	G_{FL}	directed graph $G_{FL} = (N, C)$, with N, a set of NODEs, as vertices and C, a set of CHANNELs as edges
CONTROL LAYER	G_{CL}	undirected graph $G_{CL} = (V, C)$, with V, a set of VALVEs, as vertices and C, a set of CHANNELs as edges
NODE	n	vertex in G_{FL}, physical location on the FLOW LAYER where one or more channels start or terminate
FLOW PORT	n_p	set of FLOW PORTs $N_p \subset N$, physical location where flow channels interface to locations off-chip
VALVE	v	vertex in G_{CL}, physical location where a control channel can be pressurized to create a seal over a flow channel
CONTROL PORT	v_p	set of CONTROL PORTs $V_p \subset V$, physical location where control channels interface to locations off-chip
CHANNEL	c	edge in G_{FL} and G_{CL}
CELL TRAP	c_t	set of CELL TRAPs $C_t \subset C$, physical location on a chip where cells may be placed
GATE	g	cells computing a function with $n > 0$ inputs and 1 output

port through a channel, and each control port must be connected to at least one valve. A summary of these definitions is found in Table I.

3. DESIGN FLOW

Fluigi provides the following features: logic minimization (3.1), physical design (3.2), microfluidic valve control generation (3.3), and photomask generation (3.4). Our existing control software [Thies et al. 2009] can then use the framework to perform high level simulations of chip behavior. This design flow is similar to both electronic design automation and biological design automation [Beal et al. 2012] workflows. A visual description of this flow is shown in Figure 3.

3.1. Input and Logic Minimization

Fluigi extracts the Boolean algebraic functions from an input file and converts those functions to a canonical form. It reduces the logic functions to a minimum sum of products form. This can be done using a program such as Espresso [Hayes 1993] or the Cadence Encounter design software starting from a Verilog file. We select a subset of gates (just NAND, just NOR, or a combination of AND, OR, and NOT) and represent the canonically minimized functions from the input file as functions composed exclusively of those gates. We construct a directed graph of the input functions with gates as the vertices and the connections between the gates as edges. For each gate, we select a genetic circuit that implements the functionality of that gate. Gates are duplicated to reduce fanout as molecular signals from biological components may not be concentrated enough to drive more than one input at a time. This section of our workflow is independent of the microfluidic technology used to implement the chip design.

3.2. Physical Design

The basic unit of architecture is the flow stage, which has a single signal input node, one cell sample and one media input source ports, one block of cell traps, a waste output sink port, and a single signal output node. A diagram of this is shown in Figure 4(a), and the diagram for the control layer graph and flow layer graph for such a chip is shown in Figure 4(a). The full physical design process from flow layer layout to control layer routing is shown from Figure 4(a) to Figure 4(d). We use the cell trap design

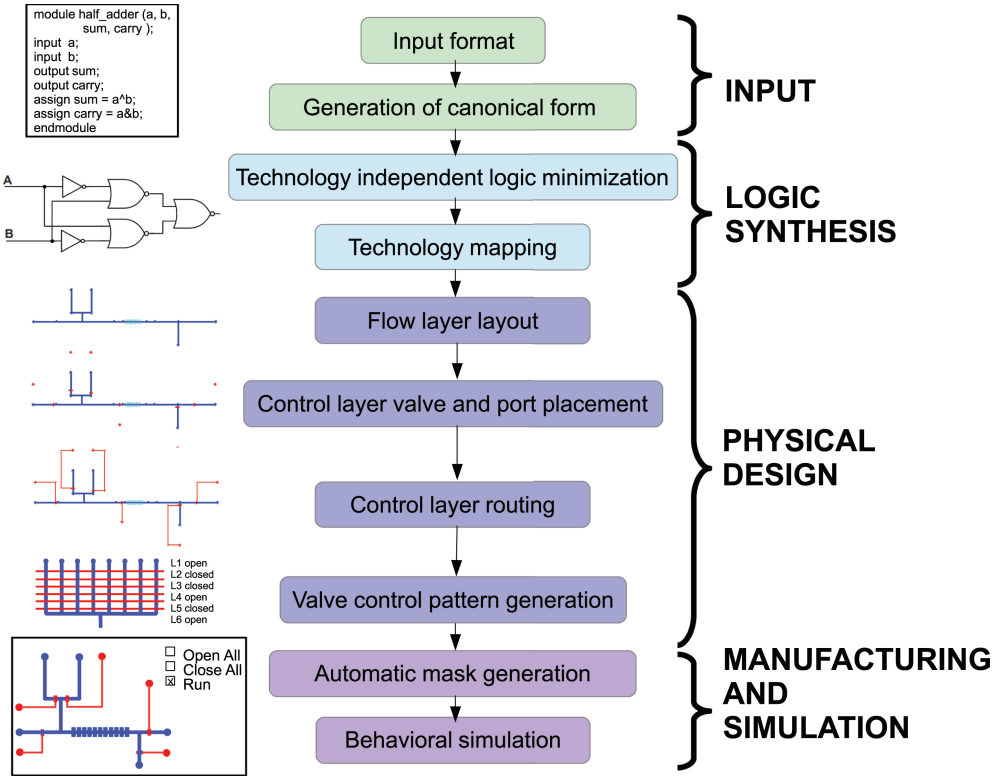


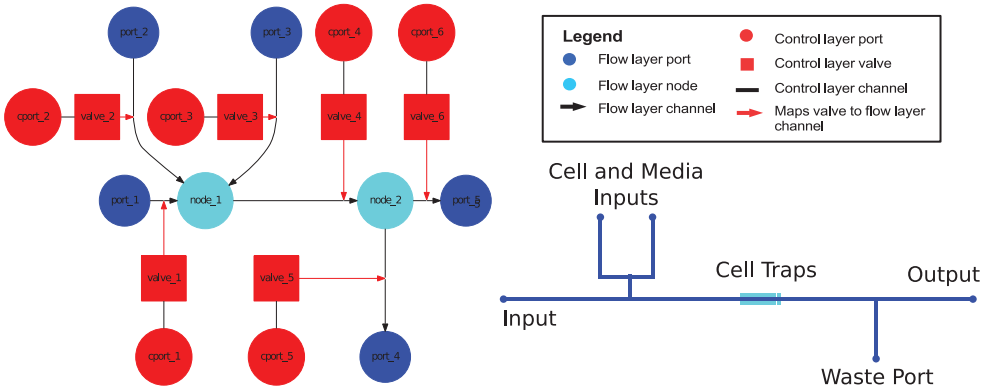
Fig. 3. Proposed design flow for Fluigi starting with an input file describing the logic function and ending with an interface to existing microfluidic control software and chip simulation.

described in Prindle et al. [2012] for our cell trap block. The flow stage unit may be expanded to contain more cell types by increasing the number of cell trap blocks and cell sample inputs. Flow stages may be linked together by connecting the output node of one stage to the input node of the next stage. This architecture gives us the flexibility to make arbitrary-sized multistage circuits.

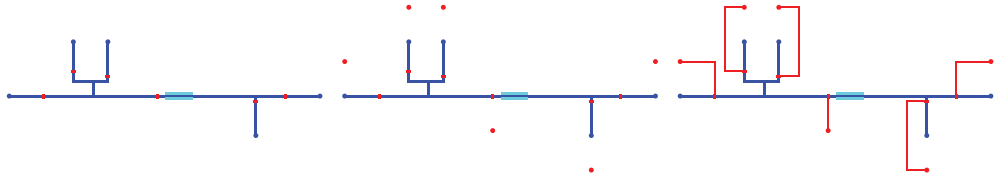
We assign valves such that each cell trap, source port, and sink port may be accessed individually and that each stage may operate independently. By connecting the cell traps and ports to be accessed in a binary tree [Thorsen et al. 2002], we can reduced the number of control lines needed to individually access those features. Using multiplexing, only $2 \log_2 n$ control lines are needed to access n samples. We apply multiplexing when the number of samples to be accessed is greater than or equal to four. Otherwise, we assign control lines individually.

During layout, we use the parameters shown in Table II to determine physical layout constraints [Amin et al. 2009] with the most important constraints being the minimum distance required around ports and the minimum spacing around channels and valves. Control valves are sized such that they can form complete seals across the channels they are meant to cover. We generate a routing grid that represents the minimum spacing needed to prevent features from violating these layout constraints and use this grid to route the channels connecting the valves and ports on the control layer.

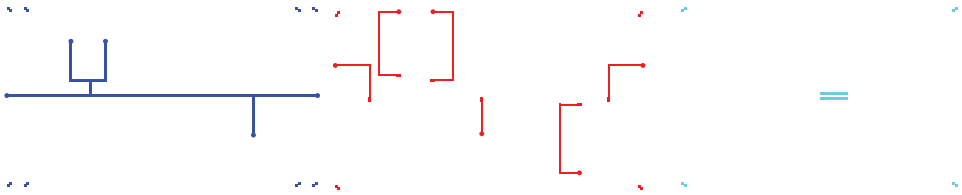
3.2.1. Flow Layer Layout. We convert the graph of gates to the graph representation of the flow layer going from the signal outputs to signal inputs. The signal flow will be



(a) Graph representation and flow layer layout of sample chip



(b) Placement of control valves (c) Placement of control ports (d) Routing of control layer



(e) Photomasks for flow layer, control layer and cell trap layer

Fig. 4. Figure 4(a) shows the graph abstraction of the sample chip. The flow layer graph is shown in blues and the control layer graph is shown in red. Dark blue circles represent flow ports, and light blue circles represent nodes. Channels in the flow layer graph are shown as black arrows going from the source node and pointing towards the target node. Red squares are valves on the control layer, and red circles are ports on the control layer. The red arrows show the flow channel that each connected set of valves and control ports is associated with. Figures 4(a) through 4(d) show the physical design process from laying out the flow layer through routing of the control layer. The main features of the primary unit of the chip architecture are the "stage", comprised of an input node, cell/media input ports, one or more cell traps, a waste port, and an output node, and are shown in Figure 4(a). The flow layer (4(a)) is laid out based on the connections present in the flow layer graph, and valves (4(b)) and ports (4(c)) in the control layer are placed to minimize routing problems. Routing the channels between valves and ports on the control layer is accomplished using the negotiated congestion algorithm. Figure 4(a) shows the photomasks generated for the sample chip for the flow, control, and cell trap layers.

from left to right, and the media and cell sample flow will be from top to bottom. We create a sink port in the flow layer for each output and then expanding leftward to create the logic stages. For each stage, we created and connect the set of cell traps, and connect the cell traps to their associated ports and the signal input and output nodes for the stage. The input node for one stage becomes the output node for the stage that precedes it. This continues until we reach the signal inputs. Here, we generate a bank of cell traps for the biological sensors meant to convert the signal of interest into the intercellular signaling chemical and the cell sample and media source ports for those cell traps. Input source ports and outputs sink ports are arranged vertically while the

Table II. Flow Layer and Control Layer Layout Parameters

Parameter	Length (μm)
Flow channel width	100
Control channel width	50
Max. control channel width when crossing flow channel	20
Flow Port Radius	100
Control Port Radius	100
Valve width	2*Flow channel width
Valve length	Flow channel width
Min. distance between ports	1000
Min. distance between port and channel	400
Min. distance between port and valve	1000
Min. distance between channels	25
Min. distance between valve and channel	25

media and cell sample source ports are arranged horizontally. The nodes in the flow layer are assigned preliminary coordinates that comply with the layout constraints, which are then converted to the nearest routing grid coordinates. The flow layer of a single stage is shown in Figure 4(a). We can convert the graph of gates to the graph of the flow layer in $O(n)$ time where n is the number of gates needed in the original logic function.

3.2.2. Control Valve and Port Placement. In general, valves on flow channels are placed near the target node in direction of fluid flow. For flow channels that begin or terminate at a flow port, we place the valve at least the minimum allowed distance away from the port. For multiplexors, we stagger the valve placement such that valves assigned to the same control line are arranged in a horizontal or vertical line depending on the flow direction of the binary tree flow channel structure. The valve placement for the sample chip is shown in Figure 4(b).

Control ports are placed to maximized routablility of the control layer. We split control port placement for multiplexors so that each line of valves have unimpeded routing access to a control port. Control ports for multiplexors containing flow ports are placed flanking those flow ports, with half the control ports on each side. For cell trap multiplexors, we align half the control ports above the cell traps and half to the below them. For single valves controlling a single flow port, we place the control port for that valve a minimum distance below the flow port. Control ports for single valves with no other associated structure are placed the minimum safe distance below the valve. The control port placement for the sample chip is shown in Figure 4(c).

3.2.3. Control Layer Routing. The routing algorithm is based on the negotiated congestion routing as presented [Ebeling et al. 1995; Betz and Rose 1997]. Each location on the routing grid where a control channel may be placed is a routing resource. As the control layer is only a single layer, the channels may not cross, and each routing resource may only be used once. The algorithm takes as an input the control layer graph and produces the list of routing resources used for each path. Each edge in the control layer graph represents a signal that must be routed. Control channels must start and end at a valve or control port, and may not otherwise intersect those features. While control channels may cross flow channels, they should only do when there is no other path, and no control channel should directly overlay a flow channel. The width of a control channel is reduced when it crosses a flow channel to lessen the chance that it would obstruct the flow. Control channels are not allowed within the minimum safe distance of any control or flow port other than the one they connect to.

There are two phases to the routing, the global routing and the individual signal routing. The signal router uses A* search to find the best path between the source and the target of the signal being routed while the global router handles the allocation of shared routing resources. In the signal router, we assess penalties for the path crossing flow channels and for having multiple bends as channels with few bends are easier to fabricate. In the first pass of the global router, paths are routed without regard to shared resources so that more than one path can use the same resource. Once all the paths have been routed, we check which paths use a resource that is used by another path. All such paths that share resources are then removed. The cost of using any routing resource that was shared on the previous iteration is then increased, and the paths that have been removed are then rerouted. This continues until there are no more shared resources. The pseudocode for this algorithm is shown in Algorithm 1, and the routed control layer for the sample chip is shown in Figure 4(d).

ALGORITHM 1: Negotiated Congestion Routing, $c_n = (b_n + h_n) * p_n$ where b_n is the base cost, h_n is the history of congestion on n in previous cycles, and p_n is related to the number of other paths using routing resource n currently.

```
while shared resources exist do
    for each edge e in the control layer graph do
        Find start and end routing resources;
        if path p exists between start and end resources && path p contains a shared resource
        then
            Clear contents of path p;
        end
        Find path p from start resource to end resource with A* search;
        for resource n in path p do
            Update penalty p_n and resource cost c_n;
        end
    end
    Check for shared resources;
    Update historical penalty h_n for all nodes;
end
```

3.3. Microfluidic Valve Control Generation

We generate Java files to interface to the existing control software BioStream, which was developed to design GUIs and control valves for multilayer devices [Thies et al. 2008; Amin et al. 2009; Urbanski et al. 2006] (freely available at Thies et al. [2009]). These files map each control line to one of the 48 physical ports on the control apparatus and define the mechanisms for opening and closing a port and for closing or opening all the ports on the chip.

We generate control patterns by first defining in the default state of the chip, when it has media flow through all cell traps to the waste ports with each stage operating independently from other stages. Valves allowing such behavior are set to the `OPEN` state, and all other valves are set to the `CLOSED` state. For all other control patterns, any valve that does not have its state set will be in the default state. We next define a set of flow paths for four possible operations in a single flow stage: loading cells and media to each cell trap individually, flushing media through all cell traps simultaneously, flushing the contents of each cell trap to the output node, and applying the inputs to each appropriate cell trap.

The number of total operations defined is based on the number of cell traps present in the flow layer. We estimate this as the number of nodes in the flow layer in the worst case. For each operation, we find the path through the control layer graph from the source port or input node to the waste port or output node for each cell trap. Each edge in the path is a channel that must remain open. If that channel has a valve placed on it, then the valve's state is set to OPEN. Path searches can be performed in $O(n)$ time with breadth first search, where n is the number of nodes in the flow layer graph. The total time to find all flow paths is then $O(n^2)$. We convert these flow paths to valve instructions in BioStream and output those instructions.

3.4. Photomask Generation

We create photomasks for manufacturing of the chip based on the layout parameters and the flow and control layer graphs. Photomasks are sued to created the master mold for the flow layer and control layer in multilayer soft lithography [Duffy et al. 1998; Sia and Whitesides 2003]. For each channel in the flow layer and control layer, we draw the channel based on the coordinates of its source and target nodes. The cell traps are drawn on a different layer than the flow channels as the cell traps will be a different depth than the channels. For the sample chip we described, we generate the three photomasks shown in Figure 4(e), as well as reference image of the three layers overlaid on each other. The reference image is used as the base for the chip level simulation.

4. RESULTS

We tested the functionality of Fluigi with two sets of benchmarks. We first constructed all the possible two- and three-input Boolean logic benchmark functions using exclusively NOR gates and ignored those with 0 inputs as those are not useful for demonstrating the capabilities of Fluigi for a total of 14 two-input Boolean functions and 254 three-input Boolean functions. The results of Fluigi on the benchmark functions using only NOR gates are shown in Table III.

The second set of benchmarks demonstrated the generalized functionality of Fluigi on more complex functions. We also tested Fluigi on four specific example circuits: (1) a 4 input AND gate, the genetic circuit of which is described in Moon et al. [2012], (2) a 3 input XOR gate, as a representative of the three-input Boolean logic function benchmarks, (3) the half adder, a simpler version of the genetic circuit described in Beal et al. [2011], and (4) an 8 to 3 encoder, the genetic circuit of which is being developed in our lab. In addition, we adapted Fluigi to describe two noncombinatorial functions, the first a microfluidic chip for constructing the repressilator [Elowitz and Leibler 2000] using three stages and a feedback channel and the second a three-stage assay similar to a neural net.

4.1. Benchmark Circuits

Under these constraints, all 14 two-input Boolean functions could be constructed using 5 stages of logic, counting the stage of input buffers used to translate the input signal to an intercellular communications signal, the two most complicated ones being the 2 input XOR and NXOR functions. These circuits were too small for multiplexing to reduce the number of control lines.

For three-input Boolean logic functions constructed with only NOR gates, the most complicated functions took 9 stages of logic, including input buffers, to carry out. Only the largest circuits with 8 or 9 stages showed any benefit of control line optimization. With these three input functions, we also notice that chips containing functions of 7 stages or more require more than 48 control lines to fully operate. Of the 254 benchmark functions, only 143 can be controlled by our existing hardware. This complication

Table III. Three-Input Boolean Logic Functions

Number of Stages	Number of Circuits	Average Unoptimized Control Lines	Average Optimized Control Lines	Average Control Line Reduction	% of Circuits Compatible with Control Setup
Built with NOR Gates					
1	3	6	6	0.00%	100
2	6	12.50	12.50	0.00%	100
3	12	20.25	20.25	0.00%	100
4	22	27.73	27.73	0.00%	100
5	74	37.45	37.45	0.00%	100
6	26	46.65	46.65	0.00%	100
7	69	58.14	58.14	0.00%	0
8	20	71.75	69.55	3.02%	0
9	22	85.95	81.95	4.54%	0
Built with AND, OR, and NOT Gates					
1	3	6	6	0.00%	100
2	15	13.4	13.4	0.00%	100
3	53	23.37	23.37	0.00%	100
4	109	34.18	34.18	0.00%	100
5	75	49.32	48.21	1.97%	61.3

is due to the minimum number of control lines needed to operate a stage. The smallest functions requiring a single stage still require six control lines to operate, which limits us to a maximum of 8 stages of single gates. Increasing the number of gates in a stage will only increase the number of control lines used from the minimum.

To mitigate this problem, we then ran all three-input benchmark circuits through Fluigi, and allowed the use of AND, OR, and NOT gates instead of relying exclusively on NOR gates. The new results for three-input benchmark functions are also shown in Table III. Allowing the use of additional gates simplifies the circuits from a maximum of 9 stages to a maximum of 5 stages. As previously, the reduction in control lines were only apparent with more complex functions. However, there still remained 29 functions that required over 48 control lines to operate. To reduce the number of control lines needed even further we would need to look at either reducing the minimum number of control lines needed in each stage, sharing control lines across stages, or better logic minimization to reduce the number of stages needed.

4.2. Example Circuits

We selected four circuits, the four-input AND gate, the three-input XOR gate, the half adder, and the 8 to 3 encoder, to describe and examine in further detail. Based on our results with the three-input benchmark functions, we use a combination of AND, OR, and NOT gates to construct these circuits to reduce the number of logic stages needed. For these four circuits, we compiled the valve control code Fluigi generated and successfully ran chip level behavior simulations of these circuits using existing microfluidics control software. The layouts for these four circuits as generated by Fluigi are shown in Figure 5, and the statistics for these are shown in Table IV.

Figures 5(c) and 5(d) show Fluigi's ability to lay out and generate controls for logic functions with more than one output. The half adder is a simpler version of the genetic arithmetic circuits first described in Beal et al. [2011] and is comprised of two smaller circuits, an AND2 gate and an XOR2 gate. Of the four sample circuits, only the three-input XOR gate and 8 to 3 decoder showed any benefit of multiplexing for control line

Table IV. Example Circuits

Circuit	Number of Stages	Average Unoptimized Control Lines	Average Optimized Control Lines	Average Control Line Reduction	Length (mm)	Width (mm)
Logic Functions						
AND4	3	26	26	0.00%	36.0	13.4
XOR3	5	54	48	11.11%	60.0	20.8
ADDER	4	41	41	0.00%	50.4	13.2
8-3 ENC3	3	48	41	14.58%	48.0	20.8
XOR4	7	97	77	20.62%	89.4	24.0
Noncombinatorial logic Functions						
Oscillator	3	17	17	0.00%	29.4	8.8
Assay	3	49	46	6.12%	54.8	17.4

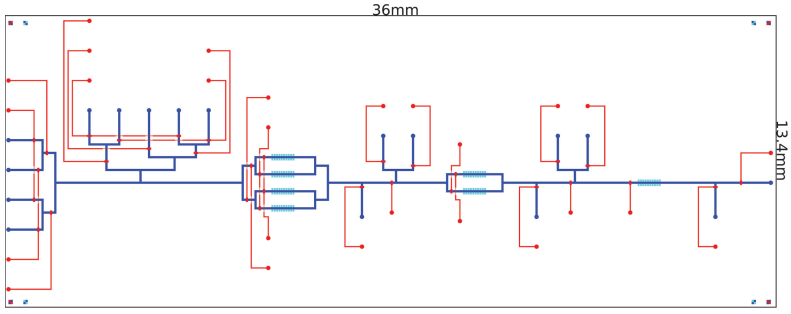
optimization for relative optimization rates of 11.11% and 14.58%, respectively. We also ran Fluigi on a 4 input XOR gate, which is a circuit larger than the biggest logic circuit constructed in synthetic biology thus far. The 4 input XOR gate is also large enough that multiplexing produces a 20.62% reduction in control line usage. However, it would still require 77 control lines to operate, and is outside the capabilities of our current hardware control system.

The benefits of multiplexing decrease as the number of cell trap blocks in each stage decrease, so multiplexing provides no benefit in later stages for circuits that only have one output. With this in mind, we then created an alternate version of input specification for Fluigi to maximize the benefits of multiplexing. We constrain Fluigi to only creating three flow stages, but allow the user to specify the number of cell trap blocks per stage. The number of inputs into the system is the number of cell trap blocks in the first stage, and the number of outputs of the system is the number of cell trap blocks in the last stage. We assume for now that each cell trap block contains a different cell strain. With this input specification type, we also allow feedback from later stages to earlier stages. We created two functions, a three-stage oscillator shown in Figure 6(a) and a three stage assay shown in Figure 6(b) with this alternate input specification to demonstrate that Fluigi is not limited to laying out logic functions. The results for these two chips are also shown in Table IV.

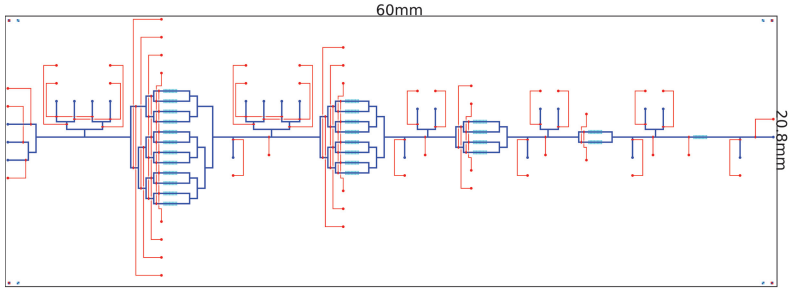
The oscillator demonstrates that our software is capable of creating feedback loops in the flow channel. In this system, each cell trap block will be loaded with a cell with a sensor for some input chemical that produces both a fluorescent protein and a chemical to inhibit the sensor in the cell in the following stage. The chip for this system has 17 control lines, with one control line per valve. As each stage is already has the minimum possible number of control lines, no further optimization can be performed. The assay design is similar to designs of neural networks, with an input layer, a hidden/processing layer, and an output layer. Since this design has multiple cell trap blocks in each layer, multiplexing allows savings of 6.12% in control line usage.

5. DISCUSSION AND FUTURE WORK

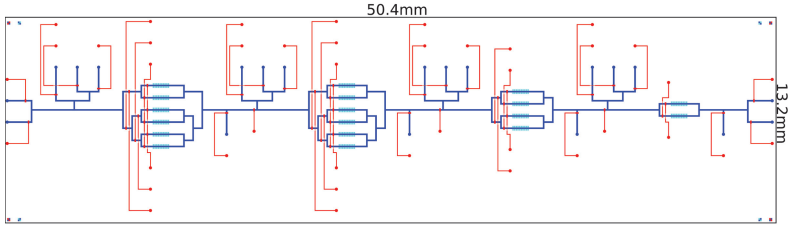
We envision combining microfluidics and biological circuits to create a biological field programmable gate array for testing of novel circuits and as a reusable experimental testbed for testing and characterization of biological devices. The architecture can also be expanded to generate a clock to synchronize biological circuits



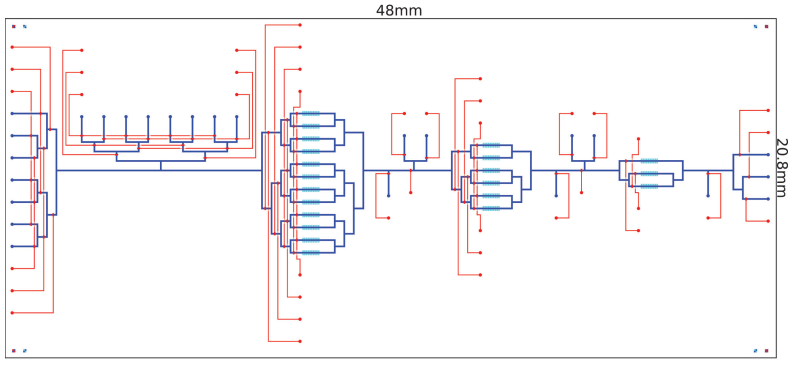
(a) AND4: 3 stages, 4 inputs, 1 output, 26 control lines



(b) XOR3: 5 stages, 3 inputs, 1 output, 48 control lines

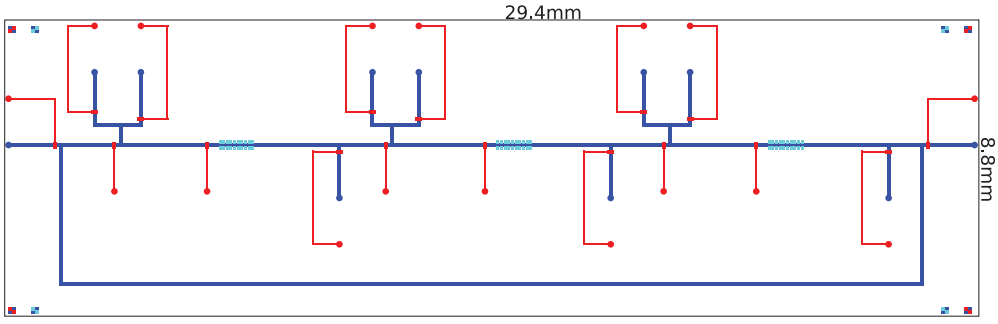


(c) Half Adder: 4 stages, 2 inputs, 2 outputs, 41 control lines

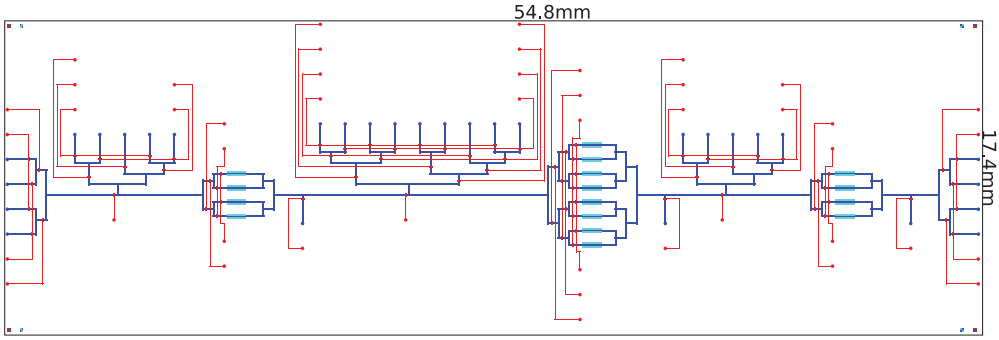


(d) 8-3 Encoder: 3 stages, 7 inputs, 3 outputs, 41 control lines

Fig. 5. Layout of selected combinatorial logic circuits as generated by Fluigi. These images serve as the GUI for simulation of high-level chip behavior and may be used to generate the photomask for chip fabrication. The sections in blue represent the flow layer and the sections in red represent the control layer. The chip dimensions are provided for each chip.



(a) 3-stage oscillator based on the repressilator



(b) 3-stage assay of 4 samples, 8 samples and 4 samples

Fig. 6. Layout of selected non-combinatorial circuits as generated by Fluigi. These images serve as the GUI for simulation of high-level chip behavior and may be used to generate the photomask for chip fabrication. The sections in blue represent the flow layer and the sections in red represent the control layer. The chip dimensions are provided for each chip.

[Prindle et al. 2012] and allow for sequential as well as combinational logic in biological devices. The next steps in our work are to use the photomasks generated by Fluigi to fabricate a physical device and test that device in the lab. The first tests will use liquid only to test connectivity and control, followed by tests with basic biological circuits to verify cell growth and signal production.

At this point, we can only hypothesize the challenges of incorporating biological circuits into our proposed microfluidic platform as we have yet to fabricate and test the microfluidic device. We base our cell trap design on the previous designs by the Hasty lab [Danino et al. 2010; Prindle et al. 2012] as we plan to use the quorum sensing system in *E. coli* as our intercellular signaling system. These cell traps have been shown to support cell growth for at least 4 days. Signal strength could be controlled by adjusting the flow rate through the cell trap. As the cell traps are 100 μm long by 100 μm wide by 1.65 μm deep, and the size of one *E. coli* cell is roughly 1 μm in diameter by 2 μm long, the cell trap will allow cell growth in a monolayer with excess cells swept into a waste channel [Danino et al. 2010]. As a result, flowing signals from one cell trap to the next after both cell traps have been filled should not cause contamination because the new cells should not have space to grow. From previous studies [Danino et al. 2010; Tamsir et al. 2011], we estimate that each stage of calculation should take on order of two hours. Given these time scales, we should be able to complete the calculations for all of our test circuits within the timeframe of cell viability. Behavior of the cells can

be monitored through time-lapse fluorescence microscopy (TLMF) [Locke and Elowitz 2009].

As the proposed design flow for the end to end workflow is modular, we could adapt the physical design portion of the workflow to work with other microfluidic technologies such as electrowetting digital microfluidics. Such work would draw heavily on the previous design automation by Chakrabarty et al. [Su and Chakrabarty 2005; Su et al. 2006; Chakrabarty and Zeng 2005; Chakrabarty 2010].

As illustrated in our results, we notice a limitation with the number of available control lines in our hardware control system. In our current design scheme, complex logic circuits, even taking advantage of multiplexing, require more control lines than the 48 lines we have. Conversely, the only designs we can test on our setup are the smaller circuits that require 48 or less control lines, which are very small compared to the current capabilities of electronic CAD. While the initial designs created by Fluigi are simple compared to electronic layouts or for biochips for biochemical assays with thousands of valves, the circuit designs we are looking at are complex for biological circuits.

Our goal is to develop a framework for quick design iteration of microfluidic devices to study distributed biological computing and the behavior of synthetic logic circuits. Microfluidic devices for studying synthetic circuit dynamics have mostly been focused on the behavior of synthetic oscillators instead of synthetic logic circuits. Those devices have not required the use of mLSI for signal control or cell isolation but rather depend on flow rate to control signal strength. On the other side of the spectrum, high-throughput screening arrays contain one to two thousand cell culture chambers and use multiplexed valves to control input into each chamber. However, these screening arrays were not designed to study multiple layers of cellular communications and could thus take better advantage of the reduction in control lines provided by multiplexing. In our case, we require the ability to control flow to and from each individual cell trap to pass the intercellular signaling chemicals. We can not reach the same level of control line optimization found in the high-throughput screening arrays because we need to individually access multiple locations on the array simultaneously.

6. CONCLUSION

We have presented here a new paradigm for combining microfluidics, logic synthesis, microfluidic physical design, and synthetic biology. We have defined a new architecture for microfluidics based on the placement of genetic circuits in discrete chambers for computation and have described an automated design flow going from a behavioral input file to a chip level behavioral simulation. We have created the CAD suite "Fluigi" as a framework for this design flow and presented the results of Fluigi on all two- and three-input Boolean functions as well as four specific example circuits. Our result show that an end-to-end workflow for microfluidic design for synthetic biology is possible and that we can introduce optimizations for control assignment and logic minimization. The integration of microfluidics and synthetic biology has the capability to increase the scale of engineered biological systems for applications in DNA assembly, biosensors, and screening assays for novel orthogonal genetic parts. Fluigi and the functions it provides represent an important step towards this integration.

ACKNOWLEDGMENTS

We would like to thank Prof. Ahmad Khalil and Brandon Wong for help with the microfluidics background, Swapnil Bhatia for his assistance with the problem definition, and Swatti Carr for her help with the genetic circuits.

REFERENCES

Nada Amin, William Thies, and Saman Amarasinghe. 2009. Computer-aided design for microfluidic chips based on multilayer soft lithography. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'09)*. IEEE, 2–9.

Ernesto Andrianantoandro, Subhayu Basu, David K. Karig, and Ron Weiss. 2006. Synthetic biology: New engineering rules for an emerging discipline. *Molec. Syst. Biol.* 2, 1.

Frederick K. Balagaddé, Lingchong You, Carl L. Hansen, Frances H. Arnold, and Stephen R. Quake. 2005. Long-term monitoring of bacteria undergoing programmed population control in a microchemostat. *Science* 309, 5731, 137–140.

Subhayu Basu, Yoram Gerchman, Cynthia H. Collins, Frances H. Arnold, and Ron Weiss. 2005. A synthetic multicellular system for programmed pattern formation. *Nature* 434, 7037, 1130–1134.

Jacob Beal, Ting Lu, and Ron Weiss. 2011. Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS One* 6, 8, e22490.

Jacob Beal, Ron Weiss, Douglas Densmore, Aaron Adler, Evan Appleton, Jonathan Babb, Swapnil Bhatia, Noah Davidsohn, Traci Haddock, Joseph Loyall, et al. 2012. An end-to-end workflow for engineering of biological networks from high-level specifications. *ACS Synthet. Biol.* 1, 8, 317–331.

Vaughn Betz and Jonathan Rose. 1997. VPR: A new packing, placement, and routing tool for FPGA research. In *Proceedings of the International Workshop on Field Programmable Logic and Appliation*.

Jerome Bonnet, Pakpoom Subsoontorn, and Drew Endy. 2012. Rewritable digital data storage in live cells via engineered control of recombination directionality. *Proc. Natl. Acad. Sci.* 109, 23, 8884–8889.

Jennifer A. N. Brophy and Christopher A. Voigt. 2014. Principles of genetic circuit design. *Nat. Meth.* 11, 5, 508–520.

Yizhi Cai, Mandy L. Wilson, and Jean Peccoud. 2010. GenoCAD for iGEM: A grammatical approach to the design of standard-compliant constructs. *Nucl. Acids Res.* 38, 8, 2637–2644.

Barry Canton, Anna Labno, and Drew Endy. 2008. Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* 26, 7, 787–793.

Krishnendu Chakrabarty. 2010. Design automation and test solutions for digital microfluidic biochips. *IEEE Trans. Circ. Syst. I* (Regular Papers), 57, 1. 4–17.

Krishnendu Chakrabarty and Jun Zeng. 2005. Design automation for microfluidics-based biochips. *ACM J. Emerg. Technol. Comput. Syst.* 1, 3, 186–223.

Deepak Chandran, Frank T. Bergmann, Herbert M. Sauro, et al. 2009. TinkerCell: Modular CAD tool for synthetic biology. *J. Biol. Eng.* 3, 1, 19.

Ramiz Daniel, Jacob R. Rubens, Rahul Sarpeshkar, and Timothy K. Lu. 2013. Synthetic analog computation in living cells. *Nature* 497, 619–623.

Tal Danino, Octavio Mondragón-Palomino, Lev Tsimring, and Jeff Hasty. 2010. A synchronized quorum of genetic clocks. *Nature* 463, 7279, 326–330.

Giovanni De Micheli. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education.

Stephan K. W. Dertinger, Daniel T. Chiu, Noo Li Jeon, and George M. Whitesides. 2001. Generation of gradients having complex shapes using microfluidic networks. *Anal. Chem.* 73, 6, 1240–1246.

David C. Duffy, J. Cooper McDonald, Olivier J. A. Schueller, and George M. Whitesides. 1998. Rapid prototyping of microfluidic systems in poly (dimethylsiloxane). *Anal. Chem.* 70, 23, 4974–4984.

Carl Ebeling, Larry McMurchie, Scott A. Hauck, and Steven Burns. 1995. Placement and routing tools for the triptych FPGA. *IEEE Trans. VLSI* 473–482.

Michael B. Elowitz and Stanislas Leibler. 2000. A synthetic oscillatory network of transcriptional regulators. *Nature* 403, 6767, 335–338.

Drew Endy. 2005. Foundations for engineering biology. *Nature* 438, 7067, 449–453.

M. S. Ferry, I. A. Razinkov, and J. Hasty. 2011. Microfluidics for synthetic biology from design to execution. *Meth. Enzymol.* 497 (2011), 295.

Ashish Gehani and John Reif. 1999. Micro flow bio-molecular computation. *Biosystems* 52, 1, 197–216.

S. Hassoun and T. Sasao. 2002. *Logic Synthesis and Verification*. Springer US. http://books.google.com/books?id=qfIFQu4o0AMC.

J. P. Hayes. 1993. *Digital Logic Design*. Addison-Wescley.

Haiyao Huang and Douglas Densmore. 2014. Integration of microfluidics into the synthetic biology design flow. *Lab on a Chip*. DOI: http://dx.doi.org/10.1039/C4LC00509K

Ahmad S. Khalil, Timothy K. Lu, Caleb J. Bashor, Cherie L. Ramirez, Nora C. Pyenson, J. Keith Joung, and James J. Collins. 2012. A synthetic biology framework for programming eukaryotic transcription functions. *Cell* 150, 3, 647–658.

Stefan Klumpp, Zhongge Zhang, and Terence Hwa. 2009. Growth rate-dependent global effects on gene expression in bacteria. *Cell* 139, 7, 1366–1375.

Benjamin Lin and Andre Levchenko. 2012. Microfluidic technologies for studying synthetic circuits. *Curr. Opin. Chem. Biol.* 16, 3, 307–317.

Wenming Liu, Li Li, Xuming Wang, Li Ren, Xueqin Wang, Jianchun Wang, Qin Tu, Xiaowen Huang, and Jinyi Wang. 2010. An integrated microfluidic system for studying cell-microenvironmental interactions versatilely and dynamically. *Lab on a Chip* 10, 13, 1717–1724.

Michael S. Livstone, Ron Weiss, and Laura F. Landweber. 2006. Automated design and programming of a microfluidic DNA computer. *Nat. Comput.* 5, 1, 1–13.

James C. W. Locke and Michael B. Elowitz. 2009. Using movies to analyse gene circuit dynamics in single cells. *Nat. Rev. Microbiol.* 7, 5, 383–392.

Timothy K. Lu. 2010. Engineering scalable biological systems. *Bioeng. Bugs* 1, 6, 378–384.

Javier Macía, Francesc Posas, and Ricard V. Solé. 2012. Distributed computation: the new wave of synthetic biology devices. *Trends Biotechnol.* 30, 6, 342–349.

Daniel Mark, Stefan Haeberle, Günter Roth, Felix von Stetten, and Roland Zengerle. 2010. Microfluidic lab-on-a-chip platforms: Requirements, characteristics and applications. *Chem. Soc. Rev.* 39, 3, 1153–1182.

Wajid Hassan Minhass, Paul Pop, Jan Madsen, and Felician Stefan Blaga. 2012. Architectural synthesis of flow-based microfluidic large-scale integration biochips. In *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. ACM, 181–190.

Wajid Hassan Minhass, Paul Pop, Jan Madsen, and Tsung-Yi Ho. 2013. Control synthesis for the flow-based microfluidic large-scale integration biochips. In *Proceedings of the 18th Asia and South Pacific Design Automation Conference (ASP-DAC'13)*. 205–212.

Tae Seok Moon, Chunbo Lou, Alvin Tamsir, Brynne C. Stanton, and Christopher A. Voigt. 2012. Genetic programs constructed from layered logic gates in single cells. *Nature* 491, 7423, 249–253.

Lior Nissim and Roy H. Bar-Ziv. 2010. A tunable dual-promoter integrator for targeting of cancer cells. *Molec. Syst. Biol.* 6, 1. DOI: http://dx.doi.org/10.1038/msb.2010.99

Arthur Prindle, Phillip Samayoa, Ivan Razinkov, Tal Danino, Lev S. Tsimring, and Jeff Hasty. 2012. A sensing array of radically coupled genetic "biopixels". *Nature* 481, 7379, 39–44.

Priscilla E. M. Purnick and Ron Weiss. 2009. The second wave of synthetic biology: from modules to systems. *Nat. Rev. Molec. Cell Biol.* 10, 6 (2009). 410–422.

Sergi Regot, Javier Macia, Núria Conde, Kentaro Furukawa, Jimmy Kjellén, Tom Peeters, Stefan Hohmann, Eulàlia de Nadal, Francesc Posas, and Ricard Solé. 2011. Distributed biological computation with multicellular engineered networks. *Nature* 469, 7329, 207–211.

Warren C. Ruder, Ting Lu, and James J. Collins. 2011. Synthetic biology moving into the clinic. *Science* 333, 6047, 1248–1252.

Sumitra Shankar and M. Radhakrishna Pillai. 2011. Translating cancer research by synthetic biology. *Molec. BioSyst.* 7, 6, 1802–1810.

Samuel K. Sia and George M. Whitesides. 2003. Microfluidic devices fabricated in poly (dimethylsiloxane) for biological studies. *Electrophoresis* 24, 21, 3563–3576.

Fei Su and Krishnendu Chakrabarty. 2005. Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips. In *Proceedings of the Symposium on Design, Automation and Test in Europe, 2005*. IEEE. 1202–1207.

Fei Su, Krishnendu Chakrabarty, and Richard B. Fair. 2006. Microfluidics-based biochips: technology issues, implementation platforms, and design-automation challenges. *Comput. Aid. Des. Integ. Circ. Syst.* IEEE Trans. 25, 2, 211–223.

Alvin Tamsir, Jeffrey J. Tabor, and Christopher A. Voigt. 2011. Robust multicellular computing using genetically encoded NOR gates and chemical "wires". *Nature* 469, 7329, 212–215.

William Thies, John Paul Urbanski, Todd Thorsen, and Saman Amarasinghe. 2008. Abstraction layers for scalable microfluidic biocomputing. *Nat. Comput.* 7, 2, 255–275.

William Thies, John Paul Urbanski, Todd Thorsen, and Saman Amarasinghe. 2009. Programable microfluidics. http://groups.csail.mit.edu/cag/biostream/. (Accessed: 2014-06-17.)

Todd Thorsen, Sebastian J. Maerkl, and Stephen R. Quake. 2002. Microfluidic large-scale integration. *Science* 298, 5593, 580–584.

Marc A. Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, and Stephen R. Quake. 2000. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science* 288, 5463, 113–116.

John Paul Urbanski, William Thies, Christopher Rhodes, Saman Amarasinghe, and Todd Thorsen. 2006. Digital microfluidics using soft lithography. *Lab on a Chip* 6, 1, 96–104.

Christopher A. Voigt. 2006. Genetic parts to program bacteria. *Curr. Opin. Biotechnol.* 17, 5, 548–557.

C. Joanne Wang, Adriel Bergmann, Benjamin Lin, Kyuri Kim, and Andre Levchenko. 2012. Diverse sensitivity thresholds in dynamic signaling responses by social amoebae. *Sci. Signal.* 5, 213, ra17.

N. H. E. Weste and D. M. Harris. 2011. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, http://books.google.com/books?id=sv8OQgAACAAJ.

Bing Xia, Swapnil Bhatia, Ben Bubenheim, Maisam Dadgar, Douglas Densmore, and J. Christopher Anderson. 2011. Developers and users guide to Clotho v2. 0 A software platform for the creation of synthetic biological systems. *Meth. Enzymol* 498, 97–135.

Received January 2014; revised June 2014; accepted August 2014