Control Synthesis for the Flow-Based Microfluidic Large-Scale Integration Biochips

Wajid Hassan Minhass, Paul Pop, Jan Madsen
Technical University of Denmark
{whmi, paul.pop, jan}@imm.dtu.dk

Tsung-Yi Ho
National Cheng Kung University
tyho@csie.ncku.edu.tw

Abstract— In this paper we are interested in flow-based microfluidic biochips, which are able to integrate the necessary functions for biochemical analysis on-chip. In these chips, the flow of liquid is manipulated using integrated microvalves. By combining several microvalves, more complex units, such as micropumps, mixers, and multiplexers, can be built. In this paper we propose, for the first time to our knowledge, a top-down control synthesis framework for the flow-based biochips. Starting from a given biochemical application and a biochip architecture, we synthesize the control logic that is used by the biochip controller to automatically execute the biochemical application. We also propose a control pin count minimization scheme aimed at efficiently utilizing chip area, reducing macro-assembly around the chip and enhancing chip scalability. We have evaluated our approach using both real-life applications and synthetic benchmarks.

I. Introduction

Microfluidics-based biochips (also referred to as lab-on-a-chip) integrate different biochemical analysis functionalities (e.g., mixers, filters, detectors) on-chip, miniaturizing the macroscopic chemical and biological processes to a sub-millimetre scale [1]. These microsystems offer several advantages over the conventional biochemical analyzers, e.g., reduced sample and reagent volumes, faster biochemical reactions, ultra-sensitive detection and higher system throughput, with several assays being integrated on the same chip [2]. Microfluidics-based biochips have become an actively researched area in recent years. These chips can readily facilitate clinical diagnostics, offer exciting application opportunities in the realm of massively parallel DNA analysis, enzymatic and proteomic analysis, cancer and stem cell research, and automated drug discovery [1, 2].

There are several types of biochip platforms, each having its own advantages and limitations [3]. In this paper, we focus on the flow-based biochips in which the microfluidic channel circuitry on the chip is equipped with chip-integrated microvalves that are used to manipulate the fluid flow [1]. A valve, shown in Fig. 1a, is the basic building block of these chips. Physically, the biochip can have multiple layers, but the layers are logically divided into two types: *flow layer* (depicted in blue in Fig. 1a) and the *control layer* (depicted in red). The liquid in the flow layer is manipulated using the control layer [1].

A valve is used to manipulate the fluid in the flow layer as the valves restrict/ permit the fluid flow. The control layer (red) is connected to an external air pressure source through a control pin z_1 (see Fig. 1a). The flow layer (blue) is connected to a fluid reservoir through a pump that generates the fluid flow. When the pressure source is not active, the fluid is permitted to flow freely (open valve). When the pressure source is activated, high pressure causes the elastic control layer to pinch the underlying flow layer (point *a* in Fig. 1a) blocking the fluid flow (closed valve). Because of their small size ($6\times6\ \mu m^2$), a biochip can easily accommodate hundreds of thousands of valves.

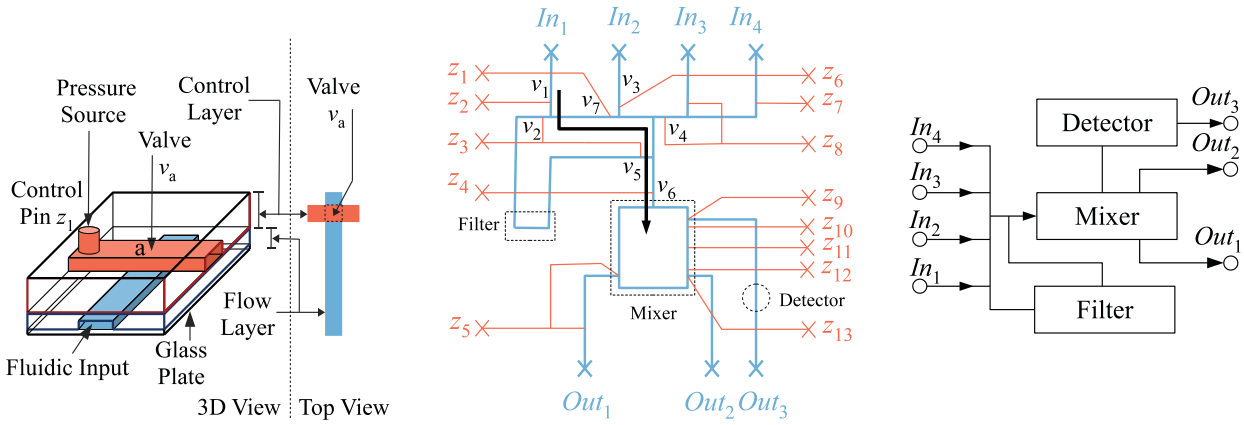
By combining several microvalves, more complex units such as mixers, micropumps, multiplexers can be built, with hundreds of units being accommodated on a single chip [3]. Analogous to its microelectronics counterpart, this approach is called microfluidic Large Scale Integration (mLSI) [1].

A. Related Work

Although biochips are becoming more complex everyday (e.g., there is a commercially available chip with more than 25,000 valves [2]), Computer-Aided Design (CAD) tools for these chips are still in their infancy. We have recently proposed a modeling and application mapping framework for these chips [4]. We have also proposed a top-down approach for synthesizing an application-specific biochip architecture, including the placement and routing approach for the physical synthesis of the flow layer [5]. A CAD tool for the control layer placement and routing has also been proposed in [6].

In this paper we target the control synthesis problem. There are two steps of control synthesis: (i) *control logic generation* and (ii) *control pin count minimization*. Control logic generation means determining which valves need to be opened or closed, in what sequence and for how long, in order to execute the application on the chip. Currently, this is done manually by the designers [7]. Generating control logic manually is inefficient, error-prone and requires the biologist (application designer) to have detailed knowledge and understanding of the chip architecture (similar to exposing the gate-level details in microelectronics). Moreover, as the chips become more complex and the assays more concurrent, the manual methodologies will not scale becoming highly inadequate.

The second step is the control pin count minimization. A *control pin* is a punch hole on the chip providing access to the control layer. In order to activate a valve, it needs to be connected to a pressure source through a control pin (e.g., in Fig. 1a, valve v_a is connected to the pressure source through control pin z_1). A valve is pressurized (closed valve) or depressurized (open valve) by the pressure source connected to the control pin. The number of valves on the chip is increasing since the manufacturing technology, soft lithography, used for the flow-based biochips has advanced faster than Moore's



(a) Microfluidic Valve (b) Biochip: Schematic View (c) Biochip: Functional View

Fig. 1.: Flow-Based Biochip Architecture Model

law [8]. Having a separate control pin for every valve directly translates to a large pin-count for the chip, resulting in (i) high consumption of the chip area, and (ii) requirement of large, expensive and complex macro-assembly around the chip (each valve requiring its own pressure source). High pin-count is a bottleneck to the scalability of these chips.

Microfluidic multiplexers [1] have previously been used to minimize the control pin count. Consider that there are N flow channels on the chip requiring N valves to close them. If every valve gets a separate pin and therefore a separate pressure source, N pressure sources will be required. However, if it is known that only one of the N flow channels needs to be open at a time (all others need to be closed), then a microfluidic multiplexer can be used. Such a multiplexer will require only $2log_2$N control pins for controlling N such flow channels, e.g., 10 pins will control 32 such flow channels. In order to perform this optimization, the user is asked to manually specify which flows on the chip will be used and if they will be used in parallel or not [6]. Once the flows have been specified, then the multiplexer is used for sharing the control pins between flow channels that satisfy the above criteria. Manual flow specification requires the user to have complete understanding of the chip as well as the application requirements. The manual input provided by the user can also easily result in under or over-constrained specifications, resulting in inefficient minimization. Moreover, the multiplexers-based minimization is only applicable if the criteria of "only one out of N flow channels needs to be open at a time" is fulfilled. The control pin count minimization problem has been shown to be NP-hard [6] but no solution or experimental results have been proposed.

B. Contribution

We propose a top-down control synthesis framework for implementing biochemical applications on flow-based biochips. Given a biochemical application modeled as a sequencing graph and a biochip architecture, control synthesis consists of the following two steps: (i) *control logic generation* and, (ii) *control pin count minimization*. We utilize the output of control logic generation step and perform the minimization by sharing the control pins between multiple valves, provided that those valves operate in unison throughout the entire application execution. Pin count minimization using multiplexers and our proposed approach can be performed one after the other in order to reach a good solution. In this paper, we do not perform multiplexing but consider that this may already have been done in the given biochip architecture.

Considering the computational complexity of the problem, we propose a Tabu Search [9] based optimization in order to minimize the pin count. Our framework does not require manual flow specifications from the user, decoupling the application design from control synthesis. To the best of our knowledge, we are the first to present an approach for the control logic generation for the flow-based microfluidic biochips.

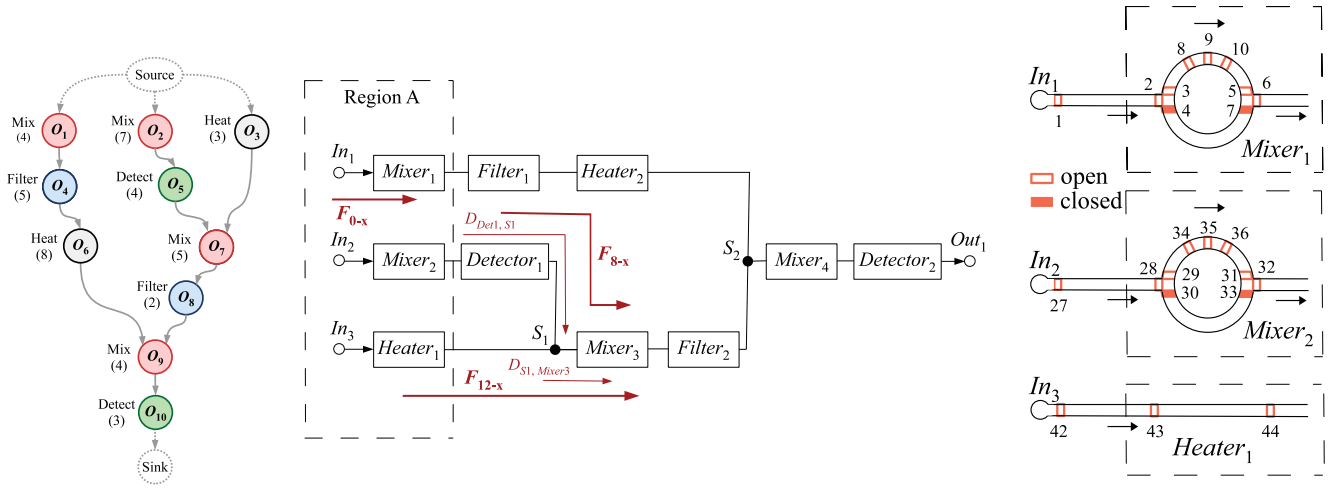
II. System Model

A. Biochip Architecture

Fig. 1b shows the schematic view of a flow-based biochip with 4 input ports and 3 output ports, 1 mixer, 1 filter and 1 detector. Fig. 1c shows the functional view of the same chip. The biochip is manufactured using multilayer soft lithography [1]. A cheap, rubber-like elastomer (e.g., PDMS) with good biocompatibility and optical transparency is used as the fabrication substrate. The path of the fluid flow is established using the microfluidic valves (e.g., for In_1 to the Mixer in Fig. 1b, $\{v_2, v_3, v_4, v_5\}$ need to be closed and $\{v_1, v_6, v_7\}$ need to be opened) and the flow is generated using off-chip or on-chip pumps.

A.1 Architecture Model

We use our previously proposed topology graph-based model [4] in order to capture the biochip architecture. The biochip architecture shown in Fig. 2b is captured by $\mathcal{A} = (\mathcal{N}, \mathcal{S}, \mathcal{D}, \mathcal{F}, \mathcal{K}, c)$, where $\mathcal{N}$ is a finite set of vertices, $\mathcal{S}$ is a set of switches (switches are formed where the channels intersect [4]), $\mathcal{S} \subseteq \mathcal{N}$, $\mathcal{D}$ is a finite set of directed edges, $\mathcal{F}$ is a finite set of flow paths and $\mathcal{K}$ is a finite set of routing constraints. Switches represent a set of valves combined together



(a) Application Graph (b) Biochip Architecture (c) Schematic View for Region A in Fig. 2b

Fig. 2.: Biochip Application and Architecture Example

and placed at the channel junctions in order to control the path of the fluid entering from different sides. A vertex $N \in \mathcal{N}$ has two types: a vertex $S \in \mathcal{S}$ represents a switch (e.g., S_1 in Fig. 2b), whereas a vertex $M \in \mathcal{N}, \notin \mathcal{S}$, represents a component or an input/output node (e.g., $Mixer_1$ and In_1 in Fig. 2b).

The set of *flow paths* $\mathcal{F}$ is the set of permissible flow routes on the biochip. A directed edge $D_{i,j} \in \mathcal{D}$ represents a directed communication channel from the vertex N_i to vertex N_j, with N_i, $N_j \in \mathcal{N}$. For example, in Fig. 2b, D_{Heater_1,S_1} represents a directed link from vertex $Heater_1$ to vertex S_1. A flow path, $F_i \in \mathcal{F}$, is either a single directed edge or a subset of two or more directed edges of $\mathcal{D}$, $F_i \subseteq \mathcal{D}$, representing a directed communication link between any two vertices $\in \mathcal{N}$. In Fig. 2b, $F_{12-x} = (D_{Heater_1,S_1}, D_{S_1,Mixer_3})$ represents a directed link from vertex $Heater_1$ to vertex $Mixer_3$. Column 1 in Table I shows the flow paths for the biochip in Fig. 2b. A shorter notation is used for describing the flow paths, i.e., instead of writing $F_{12-x} = (D_{Heater_1,S_1}, D_{S_1,Mixer_3})$, $F_{12-x} = (Heater_1, S_1, Mixer_3)$ is written. The x in F_{12-x} can have two possible values, 1 and 2. F_{12-1} represents moving the fluid from $Heater_1$ to the *upper* half of $Mixer_3$ and F_{12-2} represents moving the fluid from $Heater_1$ to the *lower* half of $Mixer_3$ (see next subsection for more details on the mixer). A routing constraint, $K_i \in \mathcal{K}$, is a set of flow paths that are mutually exclusive with the flow path $F_i \in \mathcal{F}$, i.e., none of the flow paths in the set can be activated in parallel. For example, F_{8-x} and F_{12-x} in Fig 2b are mutually exclusive as they share the vertices S_1 and $Mixer_3$. Last column in Table I shows the routing constraints for the biochip in Fig. 2b. The function $c(y)$, where y is either a directed edge $D \in \mathcal{D}$ or a flow path $F_i \in \mathcal{F}$, represents its routing latency.

Each flow path has an associated control layer model that contains the details required for its utilization, i.e., the switch and the pump activation details. Fig. 2c shows the schematic view for the area marked as *Region A* in Fig. 2b. For the flow path F_{0-1} (1 for referring to the upper half of the mixer) that goes from In_1 to $Mixer_1$, the valve set {4, 7} needs to be closed and the valve set {1, 2, 3, 8, 9, 10, 5, 6} needs to be opened (see Fig. 2c). A pumping action then moves the fluid from In_1 to the $Mixer_1$ upper half. For the biochip architecture in Fig. 2b, we consider that all the flow paths have their own dedicated off-chip pumps. The pumps can, however, be easily shared between flow paths. In that case, the flow paths that share a pump will become mutually exclusive and will be listed as such under the routing constraints. Table I shows the list of flow paths, their routing latencies and the flow path control layer models for the architecture in Fig. 2b.

A.2 Component Model

The component modeling framework is also dual-layer [4], consisting of a *flow layer model* and a *control layer model*. The flow layer model $(\mathcal{P}, C, \mathcal{H})$ of each component M is characterized by a set of operational phases $\mathcal{P}$, execution time C and the component geometrical dimensions $\mathcal{H}$. The control layer model captures the valve actuation details required for the on-chip execution of the component operation. Table II shows the flow layer model library $\mathcal{L} = M(\mathcal{P}, C, \mathcal{H})$ of seven commonly utilized microfluidic components [10].

Mixers used are pneumatic [11]. For example, $Mixer_1$ in Fig. 2c is a pneumatic mixer implemented using nine microfluidic valves, numbered 2 to 10. The valve set {8, 9, 10} acts as an on-chip pump. The valve set {2, 3, 4} and the valve set {5, 6, 7} act as switches. The two switches facilitate the inputs and outputs, and the pump is used to perform the mixing.

As shown in Table II, the mixer has five operational phases. The first two phases (Ip1, Ip2) represent the input of two fluid samples that need to be mixed (filling the upper and lower half of the mixer), followed by the mixing phase (**Mix**). The mixed sample is then transported out of the mixer in the last two phases (Op1 and Op2, emptying the two halves).

In order to perform mixing (once both halves are filled), the mixer input and output valves {2, 6} are closed while valves {3, 4, 5, 7} are opened and the mixing operation is initiated (Fig. 2c). Valve set {8, 9, 10} acts as a peristaltic pump. Closing valve 8 inserts some pressure on the fluid inside the mixer, closing valve 9 creates further pressure, then as valve 10 is

TABLE I
: Biochip Flow Path Set ($\mathcal{F}$), Control Layer Model and Routing Constraints ($\mathcal{K}$)

Flow Paths	Control Layer Model			Routing Constraints
	Flow Path	Closed Valves	Open Valves	
$F_{0-x} = (In_1, Mixer_1)$, 2 s	F_{0-1}	4, 7, 67	1, 2, 3, 5, 6, 8, 9, 10, 68	$K_{3-x} : F_{10-x}$
$F_{1-x} = (Mixer_1, Filter_1)$, 2 s	F_{0-2}	3, 5, 67	1, 2, 4, 6, 7, 68, 8, 9, 10	$K_{8-x} : F_{12-x}$
$F_2 = (Filter_1, Heater_2)$, 2 s	...			$K_{10-x} : F_{3-x}$
...	F_{6-1}	30, 33, 71	27, 28, 29, 34, 35, 36, 31, 32, 72	$K_{12-x} : F_{8-x}$
$F_{6-x} = (In_2, Mixer_2)$, 2 s	F_{6-2}	29, 31, 71	27, 28, 30, 33, 32, 72, 34, 35, 36	
...	...			
$F_{11} = (In_3, Heater_1)$, 2 s	F_{11}	64	42, 43, 44, 63	
$F_{12-x} = (Heater_1, S_1, Mixer_3)$, 3 s	...			

closed valve 8 is opened again. This forces the liquid to rotate clockwise in the mixer. The valves are closed and opened in a sequence such that the liquid rotates at a certain speed accomplishing the mixing operation. The control layer of the mixing phase is a part of the component model. Table III shows the control layer model for all components in the biochip in Fig. 2b. The control layer details are only for the functional phase of the component. For example for $Mixer_1$, the control details are for the **Mix** operation for which the valve set {2, 6} is closed, {3, 4, 5, 7} is opened and {8, 9, 10} is in the mixing state (opening and closing in a predefined sequence). Valve sequences for the heaters, filters and detectors are also given. In addition to these valve activations, the relevant component also needs to be activated, e.g., optical sensor present in the detector needs to start operation as soon as the associated valves have been activated.

The input and output phases of the components are modeled using flow paths (and their associated control layers) in the architecture model. For example, the input to $Mixer_1$ from In_1 in Fig. 2c is modeled by the flow path F_{0-x} (see Table I) and its activation is done using the associated control layer. Note that the mix valves (e.g., valve set {8, 9, 10} for $Mixer_1$) need to stay open for the input and output phases of the mixer. Mix valves are active only when the mixing is intended and need to be kept open after the mixing is done, until the desired mixed fluid has been taken out emptying the mixer.

B. Biochemical Application Model

We model a biochemical application using a sequencing graph. The graph $\mathcal{G}(\mathcal{O}, \mathcal{E})$ is directed, acyclic and polar. Fig. 2a shows an example of a biochemical application. Each vertex

TABLE II
: Component Library ($\mathcal{L}$): Flow Layer Model

Component	Phases (P)	Exec. Time (C)
Mixer	Ip1/ Ip2/ **Mix**/ Op1/ Op2	0.5 s
Filter	Ip/ **Filter**/ Op1/ Op2	20 s
Detector	Ip/ **Detect**/ op	5 s
Separator	Ip1/ Ip2/ **Separate**/ Op1/ Op2	140 s
Heater	Ip/ **Heat**/ Op	20 C/s
Metering	Ip/ **Met**/ Op1/ Op2	-
Storage	Ip or Op	-

TABLE III
: Component Control Layer Model for Fig. 2b

Component	Open Valves	Closed Valves	Mixing Valves
$Mixer_1$	3, 4, 5, 7	2, 6	8, 9, 10
$Mixer_2$	29, 30, 31, 33	28, 32	34, 35, 36
$Mixer_3$	49, 50, 51, 53	48, 52	54, 55, 56
$Mixer_4$	19, 20, 21, 23	18, 22	37, 38, 39
$Heater_1$	-	43, 44	-
$Heater_2$	-	13, 14	-
$Filter_1$	-	11, 12	-
$Filter_2$	-	57, 58	-
$Detector_1$	-	40, 41	-
$Detector_2$	-	24, 25	-

$O_i \in \mathcal{O}$ represents an operation that can be bound to a component. Each vertex has an associated weight $C_i(M_j)$, which denotes the execution time required for the operation O_i to be completed on component M_j. The execution times provided in Table II are the typical execution times for the particular component types, i.e., typical mixing time is 0.5 s but a biochemical application description may specify a longer time (e.g., 5 s) if required for a certain operation. The edge set $\mathcal{E}$ models the dependency constraints in the assay, i.e., an edge $e_{i,j} \in \mathcal{E}$ from O_i to O_j indicates that the output of O_i is the input of O_j. An operation can start when all its inputs have arrived.

III. Biochip Control Synthesis

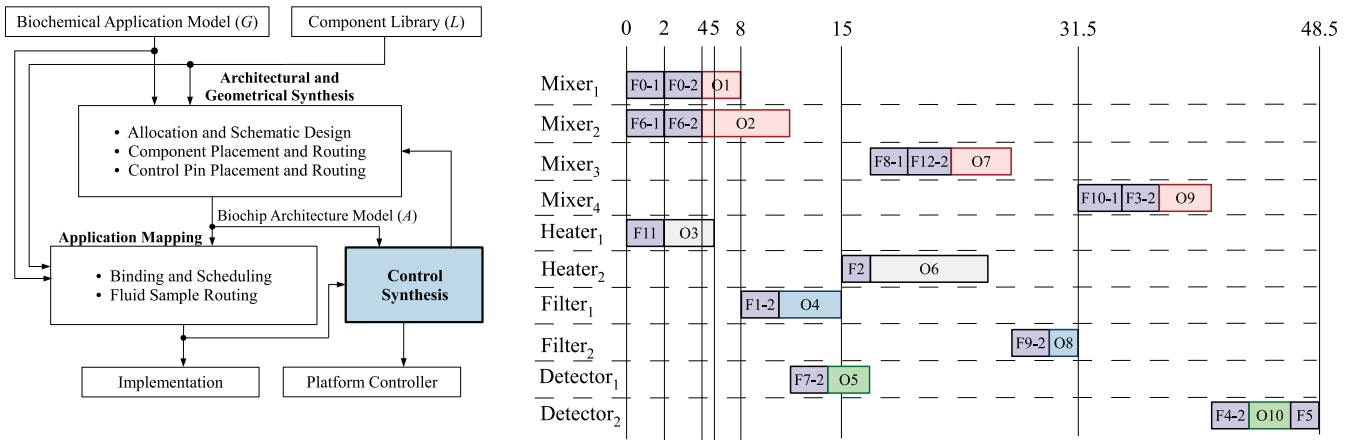
The following subsections explain the tasks involved in the biochip control synthesis using Fig. 2 as an illustrative example. Our proposed solution is discussed in Section 4.

A. Control Logic Generation

Generating the control logic η means deciding which valves to close/ open, in what sequence, at what time and for how long, in order to implement a biochemical application $\mathcal{G}$ on the chip architecture $\mathcal{A}$. It consists of the following two steps:

A.1 Application Mapping

This step consists of performing the binding and scheduling of the biochemical operations $\mathcal{O}$ onto the chip components M as



(a) Design Methodology

(b) Schedule for running the application in Fig. 2a on the biochip in Fig. 2b

Fig. 3.: Design Methodology and an Example Schedule

well as the fluid routing, i.e., binding and scheduling of the edges $\mathcal{E}$ onto the biochip flow paths $\mathcal{F}$. Fig. 3b shows the schedule $\mathcal{X}$ after mapping the application in Fig. 2a on the biochip in Fig. 2b. The schedule is depicted as a Gantt chart, where we represent the operations and fluid routing phases as rectangles, with the lengths corresponding to their execution times. The start time and the end time for the flow paths and operations in the schedule are called *time steps*, i.e., 0 s and 2 s in Fig. 3b are considered time steps since they mark the start/ end time of the flow paths, whereas, 1 s is not a time step.

A.2 Schedule Translation

The control logic η is represented in a tabular form and contains the activation status of all valves on the chip, for all time steps of the schedule. Consider the example in which the application in Fig. 2a is executed on the biochip in Fig. 2b (the schedule for this example is shown in Fig. 3b). The control logic presented in Table IV gives the activation status of the valves shown in Fig. 2c for the schedule duration 0 to 8 s.

Each row in the Table IV represents the activation status of a valve. First column contains the valve number and the remaining columns represent the activation status of the valve for the time steps present in the schedule. For example, the first row in Table IV represents the activation status of valve 1. A 0 as activation status represents an open valve, 1 a closed valve and X represents a don't-care, i.e., the valve may be opened or closed without having any influence on the application execution. For example, valve 1 (row 1 in Table IV), is opened at time 0 s, stays open at 2 s and then its status changes to a don't-care at 4 s. This is because from 0 to 4 s, F_{0-1} and F_{0-2} are executed, as shown in the schedule in Fig. 3b, filling the upper and lower halves of $Mixer_1$ (see Table I). At 4 s, both fluid samples are inside the mixer, therefore valve 2 closes in order to start the mixing operation (valve 2 status changes to 1 at 4 s in Table IV). Once valve 2 is closed, the status of valve 1 switches to a don't-care. This is because valve 1 and valve 2 are placed in series on the flow channel (see Fig. 2c) and once valve 2 is closed, opening or closing of valve 1 has no impact on the application execution. The mix valves (e.g., valve 8, 9, 10 in $Mixer_1$) act as a pump in order to achieve mixing [1]. This pumping is also included in η and for simplicity, it is shown as "Mix" in Table IV. The mix valves are opened and closed at a certain frequency in order to achieve mixing, and this opening and closing continues even between time steps.

B. Pin Count Minimization

The biochip architecture may contain some valves that are never closed during the application execution. These valves are redundant and can be removed reducing the pin count, e.g., valve 1 in Table IV is never closed and is therefore redundant. Connecting each valve to a separate control pin results in too many pin-outs from the chip limiting the chip scalability. In order to minimize the pin count, a strategy is needed to share the control pins between different valves that perform in unison with each other throughout the application execution schedule. For example in Table IV, valve 2 and valve 6 have identical activation sequence in all time steps and therefore, can share the same control pin. Similarly, valve 1 and 2 also have the same sequence (X for valve 1 at time steps 4 and 5 means that valve 1 can be switched to 1 or 0 without affecting the application execution) and can share the control pins.

C. Problem Formulation

Given (1) a biochemical application modeled as sequencing graph $\mathcal{G}$, (2) a biochip architecture modeled as a topology graph $\mathcal{A}$, and (3) a characterized component library $\mathcal{L}$, we are interested in performing the control synthesis for the given chip. Control synthesis consists of the following steps: (1) generating the control logic η needed to execute the application on the chip, while minimizing the application completion time and satisfying the dependency, resource and routing constraints, and (2) minimizing the control pin count of the chip.

IV. Synthesis Strategy

Fig. 3a shows our proposed design methodology. In this paper we focus on the "Control Synthesis" block. We consider that the biochip is given, i.e., the component and valve placement has already been done using a previously proposed tech-

TABLE IV
: Control Logic (η) Table - For Valves in Fig. 2c

Valve No.	Time Steps (s)						Color
	0	2	4	5	8	...	
1	0	0	X	X	0	...	-
2	0	0	1	1	0	...	Color - 0
3	0	1	0	0	1	...	Color - 11
4	1	0	0	0	0	...	Color - 1
5	0	1	0	0	1	...	Color - 11
6	0	0	1	1	0	...	Color - 0
7	1	0	0	0	0	...	Color - 1
8	0	0	Mix	Mix	0	...	Color - 14
9	0	0	Mix	Mix	0	...	Color - 8
10	0	0	Mix	Mix	0	...	Color - 2
...	...	...	...	...	...	...	...
27	0	0	X	X	X	...	-
28	0	0	1	1	1	...	Color - 3
29	0	1	0	0	0	...	Color - 6
30	1	0	0	0	0	...	Color - 1
31	0	1	0	0	0	...	Color - 6
32	0	0	1	1	1	...	Color - 3
33	1	0	0	0	0	...	Color - 1
34	0	0	Mix	Mix	Mix	...	Color - 9
35	0	0	Mix	Mix	Mix	...	Color - 4
36	0	0	Mix	Mix	Mix	...	Color - 7
...	...	...	...	...	...	...	...
42	0	X	X	X	X	...	-
43	0	1	1	X	X	...	Color - 13
44	0	1	1	X	X	...	Color - 13
...	...	...	...	...	...	...	...
89	X	X	X	X	X	...	Color - 4

nique (e.g., [5]). The number of control pins (punch holes in the chip providing access to the control layer) and their sharing between the valves is decided during control synthesis. The output of the "Control Synthesis" block is therefore given to the "Architectural Synthesis" block, as shown in Fig. 3a, for performing the control pin placement and control channel routing, connecting the valves to the control pins. Fig. 4 shows our control synthesis algorithm.

A. Control Logic Generation

A.1 Application Mapping

We use our previously proposed approach in [4] for this task (line 2 in Fig. 4). Our approach uses List Scheduling to perform the binding and scheduling of operations and to perform the fluid routing. Schedule $\mathcal{X}$ in Fig. 3b has been obtained by mapping the application in Fig. 2a on to the architecture in Fig. 2b using our proposed approach [4]. In our design methodology in Fig. 3a, this is done in a separate block "Application Mapping" and then the output is given to the "Control Synthesis" block.

A.2 Schedule Translation

Control logic η is generated by fetching the control layer model of the biochip flow paths $\mathcal{F}$ and components M (part of the biochip architecture model $\mathcal{A}$), and utilizing them to translate

ControlSynthesis($\mathcal{G}$, $\mathcal{A}$, $\mathcal{L}$, maxIter)

```
1  // Map application to generate schedule
2  X = MapApplication(G, A, L)
3  // Generate control logic
4  η = GenCtrlLogic(F, M, X)
5  // Algorithm for pin count minimization
6  Gc = GenGraph(η)
7  k = |Gc|, kIterate = 1
8  while f(s) > 0 and kIterate = 1 do
9     // Generate initial solution
10    s = Random(Gc)
11    Initialize TL to ϕ, nrIter = 0, s* = s
12    while f(s) > 0 and nrIter < maxIter do
13       BestMove(s') = GenNbrSelectMove(s, TL)
14       Update TL
15       if f(s*) > f(s') then
16          s* = s', nrIter = 0
17       else
18          nrIter++
19       end if
20       PerformMove {s = s'}
21    end while
22    (k, kIterate) = BSearch(f(s), k)
23 end while
24 return <η, G_C, s*, k>
```

Fig. 4.: Synthesis Algorithm

the schedule $\mathcal{X}$ into the valve activation sequence (line 4 in Fig. 4). At every time step of the schedule $\mathcal{X}$ (generated in the previous step), we look at the active flow paths and operations, fetch the associated control layer models and populate the table representing the control logic. The valves that need to be opened are given a status 0, the ones that need to be closed 1 and to the set of valves that are mixing the status "Mix" is allotted. All other valves are set as X (don't-care) for this time step and then the algorithm moves on to the next time step. For example at time step 2, operation O_3 and flow paths F_{0-2}, F_{6-2} are active, as shown in the schedule in Fig. 3b. Operation O_3 is bound to $Heater_1$, so we fetch the control layer model for $Heater_1$ from Table III according to which valves 43, 44 should be closed. The status for these valves is thus set to 1 at time step 2 in the control logic (Table IV). Similarly the control layer models for the flow paths F_{0-2} and F_{6-2} are fetched from Table I and the valves involved are set to either 1 or 0, depending on whether they needed to be closed or opened. All other valves (except the mix valves) are set to the status X.

The mix valves are assigned a don't care status X only when either both halves of the mixer are empty, or when the mixed fluid in only one half of the mixer was required for the application and that half has been emptied. When mix valves (e.g., {8, 9, 10} for $Mixer_1$) are set to X, the input and output valves of the respective mixer ({2, 6} for $Mixer_1$) need to be closed. This ensures that if these mix valves (e.g., {8, 9, 10} of $Mixer_1$) share control pins with other mix valves (e.g., {34, 35, 36} for $Mixer_2$) and a pumping action is performed because of this, the pumping affect is contained inside the mixer and does not affect the rest of the chip operation.

B. Pin Count Minimization

The pin count minimization problem has previously been reduced to a graph coloring problem (GCP) [6]. In GCP, the nodes in the graph need to be colored using minimum number of colors, in such a way that no two adjacent nodes have the same color. Finding the exact chromatic number (the minimum number of colors that can be used to color the graph nodes) is an NP-hard problem [6]. Finding out if the graph can be covered with k colors is an NP-complete problem [9]. Although the problem has previously been reduced to a GCP, however, no solution or experimental results have been proposed.

B.1 Graph Generation

Before we generate the graph, we remove redundant valves, if any, from the biochip architecture. Redundant valves are the ones that are never closed during the entire application execution, e.g., valve 1, 27 and 42 in Table IV are redundant valves as their status is never set to 1. These valves can be removed from the chip architecture as their presence has no effect on the application execution.

Next, we create the graph $\mathcal{G}_C(\mathcal{V}_C, \mathcal{E}_C)$ (line 6 of Fig. 4) by considering each valve in Table IV as a separate node Vc in the graph (redundant valves are not considered). An edge Ec is made between two nodes if a time step exists in the schedule for which the valves (represented by the nodes) have a different activation status. For example, the nodes representing valve 2 and valve 6 will not have an edge between them as they operate in unison throughout the schedule as shown in Table IV, but an edge will be made between valve 2 and 3 since their activation status vary at time step 2 (valve 2 is open and valve 3 is closed). The graph is complete once all edges have been drawn. The graph for Table IV has 83 nodes (total valves were 89, 6 were found to be redundant and were removed) and 1312 edges.

B.2 Pin Count Minimization Algorithm

The problem for pin count minimization is now represented in the form of a classical graph coloring problem (GCP). Once the colors have been assigned, the nodes that have the same color will share the same control pin.

GCP has been studied extensively and different approaches have been proposed to find a good quality solution. The simplest approach is the Greedy method [9] which takes the ordering of the nodes as input and colors them with the smallest color number, while satisfying the constraint that no two adjacent nodes should have the same color. Greedy often performs poorly in practice since a good node ordering is difficult to decide. DSATUR is another commonly utilized approach that uses a heuristic to dynamically change the ordering of the nodes and then uses the Greedy method for coloring [9]. Branch and bound algorithms [9] and semi-definite programming solutions for approximate graph coloring [12] have also been proposed. Considering the complexity of the problem, different metaheuristic techniques have also been used extensively for finding good graph coloring solutions, especially when there is a large number of nodes [9]. We use a Tabu Search-based optimization scheme in order to perform the pin count minimization.

Tabu Search (TS) is a metaheuristic based on a neighbourhood search technique which uses design transformations (moves) applied to the current solution in order to generate a set of neighbouring solutions that can be further explored by the algorithm [13, 14]. Our algorithm aims to target a k-GCP, i.e., finding a graph coloring using k number of colors and then iterates to find the smallest value of k.

Given the graph $\mathcal{G}_C(\mathcal{V}_C, \mathcal{E}_C)$ to be colored, the target of the algorithm is to partition the nodes into a fixed k number of subsets, such that no two adjacent nodes belong to the same subset. We start with a random solution s for the k-coloring (line 10 in Fig. 4), which (typically) contains a high number of edges in a conflict, i.e., the nodes connected by these edges have the same color. Using this as an initial solution the algorithm iterates to explore the solution space S, which is the set of all possible k-colorings of the graph.

In order to efficiently perform the search, TS uses memory to record moves that are not allowed at the present iteration (called tabu list (TL)). The aim is to exclude moves that would cycle the search back to the local optima that has already been evaluated. A move remains tabu only for a certain number of iterations, this is called the size of tabu list. Our algorithm starts with an empty TL and the best known solution $s*$ is initialized with the initial solution s (line 11).

Next, the algorithm generates neighbourhoods for the current solution s and picks the best one from these based on a certain objective function (line 13). A neighbour s' is generated by selecting a random node x that is adjacent to an edge in conflict. Assuming $x \in V_{Ci}$, we choose a random color $j \neq i$ and replace i with j to obtain s'. The objective function $f(s)$ measures the number of edges in conflict:

$$f(s) = \sum_{i=1}^{k} |E_{Ci}| \tag{1}$$

where $s \in S$ is the solution under evaluation, and E_{Ci} is the set of edges in conflict, having color i. The objective is to determine a k-coloring such that $f(s) = 0$. The best move is the one that is not tabu and has the lowest $f(s)$ value.

Our algorithm generates the tabu list as follows: whenever a node x is moved from V_{Ci} to V_{Cj}, the pair (x, i) becomes tabu, i.e., node x cannot be returned to V_{Ci} for a certain number of iterations. However, in order to not prohibit attractive moves, our algorithm uses the aspiration criteria of allowing the tabu moves if they result in a solution s' that is better than the currently known best solution $s*$. The best move is selected (line 13) and the tabu list is updated (line 14).

The algorithm starts with the value of k equal to the number of nodes in the graph (line 7). For this k, the algorithm continues searching until it finds a solution for which $f(s) = 0$ or until it reaches the maximum number of iterations without an improvement in the solution (number of iterations, nrIter, is reset and incremented in line 16 and 18 respectively). Our algorithm uses binary search (half-interval search) to select new value of k, as it iterates to find the minimum value of k for which $f(s) = 0$ (line 22). When the minimum k is found, the algorithm stops and the control logic η and the pin sharing information ($\mathcal{G}c$, $s*$, k) is returned (line 24). The graph generated for Table IV has 83 nodes, i.e., 83 valves requiring 83 con-

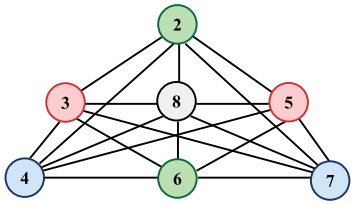


Fig. 5.: Colored Graph

trol pins and therefore 83 off-chip pressure sources. After performing the pin count minimization, only 15 control pins were found to be needed to control these 83 valves and to execute the control logic given in Table IV. Last column in Table IV shows the colors assigned to the valves (valves with the same color share the control pin). A small section of the colored graph is shown in Fig. 5 representing the pin sharing, e.g., valve 2 and valve 6 do not have an edge between them, hence they share the same color (and therefore the same control pin).

V. Experimental Evaluation

We evaluate our proposed approach by synthesizing real-life assays as well as a set of synthetic benchmarks. The algorithm was implemented in C++, running on Lenovo T400s ThinkPad with Core 2 Duo Processors at 2.53 GHz and 4 GB of RAM.

Table V shows the results. Column 1 shows the application name (for a real-life assay) or the number of operations in the application (in case of a synthetic benchmark). The second column gives details of the biochip architecture in the following format: (Mixers, Heaters, Filters, Detectors). The third column shows the number of control pins originally needed and column 4 shows the number of control pins needed once the redundant valves have been removed (the term NR in the table means Non-Redundant). Column 5 gives the final pin count after the minimization has been performed.

The first real-life assay is PCR (polymerase chain reaction) mixing stage that has 9 mix operations and is used in DNA amplification [15]. Next, real-life assay Multiplexed IVD (in-vitro diagnostics) has a total of 12 operations and is used to test fluid samples from the human body. The synthetic benchmark applications are composed of 10 up to 50 operations. In all cases, the pin count is reduced by more than 70%, e.g., for the 10 node application it goes down from 87 to 15, an 82% reduction in pin count. The parameter values were determined by tuning. Size of TL was set to 7 and the number of neighbours/ sample to 15.

TABLE V
: Experimental Results

Appl./ Nodes	Allocated Units	Total Pins	NR Pins	Optimized Pins
PCR	(4, 0, 0, 0)	73	52	12
IVD	(6, 0, 0, 6)	65	65	12
10	(4, 2, 2, 2)	87	83	15
20	(12, 4, 3, 1)	164	145	38
30	(17, 6, 4, 3)	212	201	47
40	(21, 9, 6, 4)	269	197	74
50	(26, 12, 7, 5)	329	240	61

The algorithm execution time depends on the chosen value of k and the size of the problem. On average, for each value of k, the algorithm iterates for 2 to 4 minutes. All architectures and benchmarks can be found here [15].

VI. Conclusions

In this paper, for the first time to our knowledge, a top-down control synthesis framework for implementing the biochemical applications on the flow-based biochips has been proposed. Our algorithm generates the control logic needed to execute the application and uses a Tabu Search-based optimization in order to minimize the control pin count. The minimization is targeted at efficiently utilizing the chip area, reducing the macro-assembly around the chip and enhancing scalability of mLSI biochips. The framework has been evaluated using real-life applications as well as a set of synthetic benchmarks.

VII. Acknowledgments

This work was supported by the Danish Agency for Science, Technology and Innovation, Grant 2106-08-0018 “Pro-Cell” and the Taiwan National Science Council, Grant NSC 101-2220-E-006-016 and NSC 101-2628-E-006-018-MY3.

References

[1] J. Melin and S. Quake, “Microfluidic large-scale integration: The evolution of design rules for biological automation,” *Annual Reviews in Biophysics and Biomolecular Structure*, vol. 36, pp. 213–231, 2007.

[2] J. M. Perkel, “Microfluidics - bringing new things to life science,” *Science*, November 2008.

[3] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, “Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications,” *Chem. Soc. Rev.*, vol. 39, pp. 1153–1182, 2010.

[4] W. H. Minhass, P. Pop, and J. Madsen, “System-level modeling and synthesis of flow-based microfluidic biochips,” in *Proceedings of the International Conference on Compilers, Architectures and Synthesis of Embedded Systems (CASES)*, 2011, pp. 225–233.

[5] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, “Architectural synthesis of flow-based microfluidic large-scale integration biochips,” in *Proceedings of the International Conference on Compilers, Architectures and Synthesis of Embedded Systems (CASES)*, 2012, pp. 181–190.

[6] N. Amin, W. Thies, and S. Amarasinghe, “Computer-aided design for microfluidic chips based on multilayer soft lithography,” in *Proceedings of the IEEE International Conference on Computer Design*, 2009.

[7] W. B. Thies, “Programmable microfluidics,” *presented at Stanford University*, October 2007.

[8] J. W. Hong and S. R. Quake, “Integrated nanoliter systems,” *Nature Biotechnology*, vol. 21, pp. 1179–1183, 2003.

[9] A. Lim, Y. Zhu, and Q. Lou, “Heuristic methods for graph coloring problems,” in *ACM symposium on Applied Computing*, 2005, pp. 933–939.

[10] Y. C. Lim, A. Z. Kouzani, and W. Duan, “Lab-on-a-chip: a component view,” *Journal of microsystems technology*, vol. 16, no. 12, 2010.

[11] H. Chou, M. Unger, and S. Quake, “A microfabricated rotary pump,” *Biomedical Microdevices*, vol. 3, 2001.

[12] D. Karger, R. Motwani, and M. Sudan, “Approximate graph coloring by semidefinite programming,” *Journal of the ACM (JACM)*, vol. 45, pp. 246–265, 1998.

[13] F. Glover and M.Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997.

[14] A. Hertz and D. Werra, “Using Tabu search techniques for graph coloring,” *Journal of Computing*, vol. 39, pp. 345–351, 1987.

[15] “mLSI Biochips,” https://sites.google.com/site/mlsibiochips/.