Pin Count-Aware Biochemical Application Compilation for mVLSI Biochips

Michael Lander Raagaard and Paul Pop
Technical University of Denmark, DK-2800 Kgs. Lyngby
email: paupo@dtu.dk

***Abstract* - Microfluidic biochips are replacing the conventional biochemical analyzers and are able to integrate the necessary functions for biochemical analysis on-chip. In this paper we are interested in flow-based biochips, in which the fluidic flow manipulated using integrated microvalves, which are controlled from external pressure sources, connected to "control pins". By combining several microvalves, more complex units, such as micropumps, switches, mixers, and multiplexers, can be built. The current practice is to design these biochips by hand in drawing tools such as AutoCAD, and to program them manually by individually controlling each valve. Recent research has proposed top-down physical synthesis Computer-Aided Design tools, and programming languages and compilation techniques to automatically derive the control signals for the valve actuations. However, researchers have so far assumed that the number of ports used to drive the valves (control pins) is unlimited, which has resulted in very expensive, bulky and energy consuming off-chip control and infeasible control routes in the biochip control layer. In this paper, we propose a methodology to reduce the number of control pins required to run a biochemical application. We focus on the compilation task, where the strategy is to delay operations, without missing their deadlines, such that the sharing of control signals is maximized. The evaluation shows a significant reduction in the number of control pins required.**

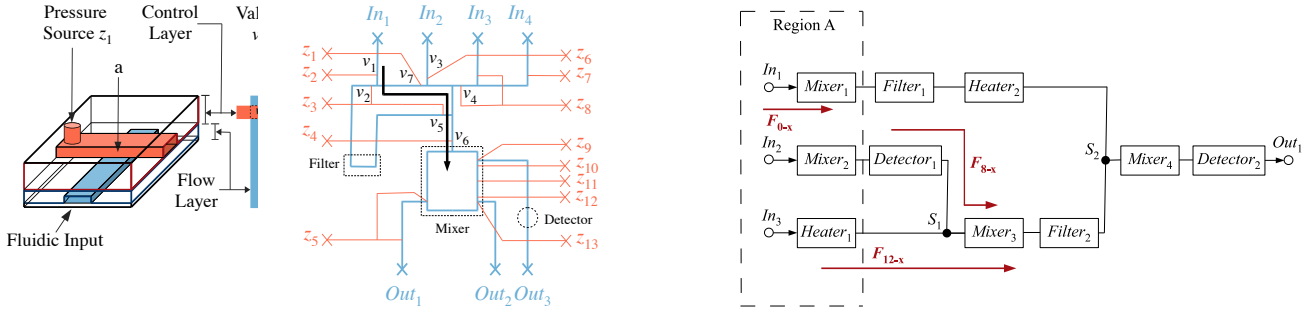
I. Introduction

Microfluidics-based biochips have become an actively researched area in recent years. Biochips integrate different biochemical analysis functionalities (e.g., dispensers, filters, mixers) on-chip, miniaturizing the macroscopic chemical and biological processes to a sub-millimeter scale [1]. Microfluidic biochips can readily facilitate clinical diagnostics, enzymatic and proteomic analysis, cancer and stem cell research, and automated drug discovery [2]. There are several types of microfluidic biochip platforms, each having their own advantages and limitations [8]. In this paper, we focus on the flow-based biochips, which use microvalves to manipulate the on-chip fluid flow [1]. By combining several microvalves, more complex components, such as, mixers, micropumps, multiplexers etc., can be constructed, with hundreds of units being accommodated on one single chip. This approach is called *microfluidic Very Large Scale Integration* (mVLSI) [1].

Such mVLSI biochips can be complex, for example, an mVLSI biochip with over 25,000 valves has been demonstrated [2], and the current practice is to expose all of the biochip components to the biochemist end-user, who has to control individually each component (e.g., a valve) to implement the protocols. This is like programming computers by toggling individually each transistor.

To address this challenge, researchers have proposed languages such as BioCoder [5] and Aqua [6] (a proprietary language from Microfluidic Innovations, LLC) and compilation techniques, which, starting from a biochemical application, derive the control signals (air pressure) to the valves. Researchers have so far assumed that the number of ports used to drive the valves (*control pins*) is unlimited, and have proposed compilation techniques to synthesize the control signals [3]. However, the number of control pins is limited in practice, hence researchers have proposed methods to minimize the number of required control pins to run the application [7, 13]. Because these methods are applied after the control signals have been synthesized, they have a limited opportunity to reduce the number of control pins.

In this paper we propose, for the first time to our knowledge, a pin count-aware compilation technique that aims to reduce the number of control pins during the operation binding and scheduling steps. This is achieved by delaying operations, without missing their deadlines, such that the sharing of control signals is maximized. The evaluation shows a significant reduction in the number of control pins required. The results have been validated on the Microfluidics Innovation (MI) platform [6], which has 30 control pins.



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

Fig. 1. Flow-based Biochip Architecture Model

II. System Model

A. Biochip Architecture Model

Fig. 1b shows the schematic view of a flow-based biochip with four input ports and three output ports, a mixer, a filter and a detector. The control pins are marked with a red "x" symbol and are labeled with z_i. Fig. 1c shows the functional level view of another chip, which we will use as an example. An mVLSI biochip is manufactured using multilayer soft lithography [1]. A cheap, rubber-like elastomer (polydimethylsiloxane, PDMS) with good biocompatibility and optical transparency is used as the fabrication substrate. Physically, the biochip can have multiple layers, but the layers are logically divided into two types: *flow layer* (depicted in blue) and the *control layer* (depicted in red). The liquid in the flow layer is manipulated using the control layer.

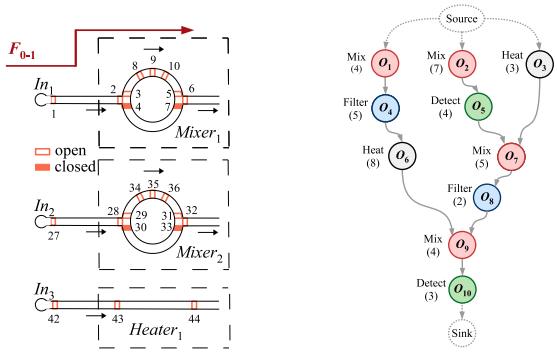
The basic building block of such a biochip is a valve (see Fig. 1a), which is used to manipulate the fluid in the flow layer as the valves restrict/permit the fluid flow. The control layer (red) is connected to an external air pressure source. The flow layer (blue) is connected to a fluid reservoir through 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 at point a (closed valve).

The component modeling framework consists of a *flow layer model* and a *control layer model*. The flow layer model of each component is characterized by a set of operational phases, execution time and the component geometrical dimensions. The control layer model captures the valve actuation details required for the on-chip execution of the component operation. The following table shows the flow layer model library of seven commonly utilized microfluidic components:

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	-

For example, $Mixer_1$ in Fig. 2a 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 the table, 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, and then as valve 10 is 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.



(a) Region A in Fig. 1c (b) Application model example

Fig. 2. Schematic view and application example

The pressure to the microvalves is delivered through off-chip *control pins*, which connect the pressure sources, via the control channels, to the microvalves. The pressure is turned on and off via off-chip expensive and bulky solenoid valves, which are in turn controlled by a computer. Fig. 4c shows an mVLSI biochip control system from Microfluidics Innovation, LLC, which has 30 pressure control pins, interfaced to the biochip through a manifold.

B. Biochemical Application Model

We model a biochemical application using a sequencing graph [4]. Such a graph can be obtained by processing the source code of a high-level language, e.g., BioCoder or Aqua. Each vertex 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 the previous table 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. An edge 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. Fig. 2b shows an example of a biochemical application. We assume that the designer has specified a deadline D for the application, which is the latest time when the application has to complete its execution.

IV. Control Pin Minimization Strategy

The overall strategy for control pin minimization is presented in Fig. 3, and has 5 steps. The focus of this paper is on step 1 "Compilation". The related work for step 1 was covered in the introduction, and the related work for steps 2 to 5 is mentioned during the discussion of the respective step.

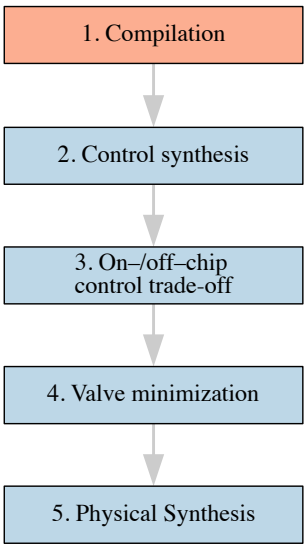


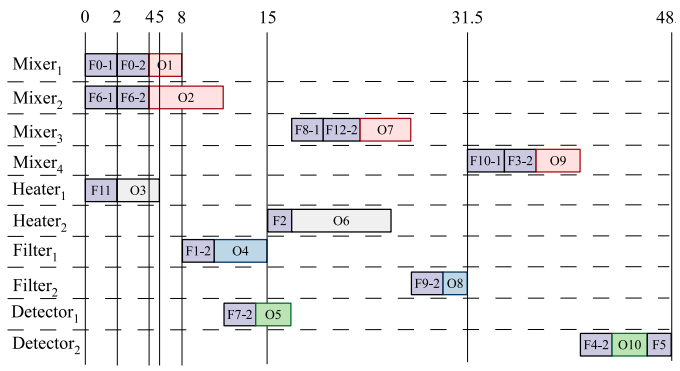
Fig. 3 Overview of the minimization strategy

(1) As a first step, we perform a *compilation* of the biochemical application. The compilation step takes as input the biochemical application model (see Section II.B), which is derived from the biochemical protocol, expressed in natural language or in a high-level language such as Aqua or BioCoder. We are currently working on software to automatically derive a sequencing graph from Aqua source code. The compilation step also requires as input the biochip architecture model on which the application has to run, see Section II.A. The compilation process is a NP-complete problem, which, consists of the following tasks: *resource binding*, *scheduling* and *fluid routing*. The output of the compilation is a schedule table. For the example biochip architecture model in Fig. 1c, and the example application model in Fig. 2b, one possible schedule table is depicted in Fig. 4a. Section IV.C presents our pin-count aware compilation approached, aiming at reducing the number of control pins.

(2) In the second step, we perform *control synthesis*. Starting from a schedule produced in the previous compilation step, we derive the control logic η, which 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. 2b is executed on the biochip in Fig. 1c (the schedule for this example is shown in Fig. 4a). The control logic presented in Fig. 4b gives the activation status of the valves shown in Fig. 2a for the schedule duration 0 to 8 s. Each row in the Fig. 4b 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 Fig. 4b 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 Fig. 4b), 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. 4a, filling the upper and lower halves of $Mixer_1$. 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 Fig. 4b). 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 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. This pumping is also included in η and for simplicity, it is shown as "Mix" in Fig. 4b. 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.

(3) *On-/Off-chip control trade-off*. As mentioned, the microvalves are controlled by pressure. A channel in the control layer is connects a control pin to the microvalve. Flexible tubing connects the off-chip solenoid valves to the control pins. See Fig. 4c for a setup where 30 solenoid valves (in the white box behind the biochip) are connected to 30 control pins, which in turn control 30 microvalves. So far, all the biochip control has been performed "off chip", i.e., done by controlling the off-chip solenoid valves attached to the control pins. This has created a situation where instead of having a "lab-on-a-chip" we actually have a "chip-in-a-lab", connected to off-chip systems through a maze of tubing. This off-chip control is expensive, bulky, and power hungry, and hence is an obstacle to the use of biochips in personal diagnostics, emerging economies, etc.

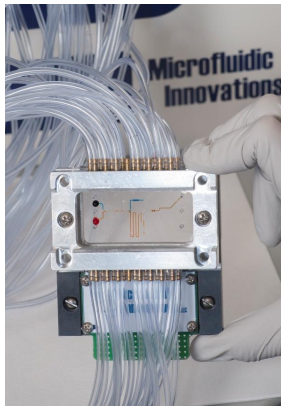
As a solution to this problem, researchers have started to propose on-chip control using pneumatics, and have proposed multiplexers (see Fig. 6c, where with 8 pressure inputs we can drive 16 pressure outputs, reducing thus the number of control



(a) Schedule for the application in Fig. 2b running on the biochip in Fig. 1c

Valve No.	Time Steps (s)					
	0	2	4	5	8	...
1	0	0	X	X	0	...
2	0	0	1	1	0	...
3	0	1	0	0	1	...
4	1	0	0	0	0	...
5	0	1	0	0	1	...
6	0	0	1	1	0	...
7	1	0	0	0	0	...
8	0	0	Mix	Mix	0	...
9	0	0	Mix	Mix	0	...
10	0	0	Mix	Mix	0	...
...	...	...	...	...	...	...
27	0	0	X	X	X	...
28	0	0	1	1	1	...
29	0	1	0	0	0	...
30	1	0	0	0	0	...
31	0	1	0	0	0	...
32	0	0	1	1	1	...
33	1	0	0	0	0	...
34	0	0				..
35	0	0	89 valves			..
36	0	0	485 time steps			..
...	...	...	...	...	...	...

(b) Control signals to valves



(c) MI mVLSI biochip [6]

Fig. 4. Schedule and associated control signals to the valves

pins required), pneumatic transistors, memory latches, logic gates, shift-registers, adders, and even clocks and finite-state machines [9, 10, 11]. The vision is to move all the off-chip control on-chip, removing completely the dependence on external control. A more realistic scenario is to find the right trade-off between off-chip and on-chip control, depending on the constraints imposed by the designer. For example, if the off-chip control already has 30 pressure ports (such as the control box from MI), then only the rest of the control signals would be moved on-chip. Another constraint can be the on-chip area available for on-chip control, which can take significant space. We see this as an interesting on/off-chip control trade-off problem. At the moment this trade-off and the design of the on-chip control are performed manually. In our future work we plan to develop computer-aided design tools, which can automatically decide on the right trade-off between off- and on-chip control (depending on the constraints provided by the designer) and synthesize the on-chip control.

(4) *Valve 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 Fig. 4b 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 Fig. 4b, 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. We discuss a possible solution to the valve minimization step in Section IV.B.

(5) *Physical synthesis*. Motivated by the similarity between VLSI and mVLSI, we have proposed an mVLSI design flow [12, 16, 18, 20], and have developed tools for the physical design of biochips. Given the system specifications (e.g., application requirements, chip area), the mVLSI design flow starts with the schematic design of the required biochip. This is followed by the physical synthesis of the flow layer, i.e., placement of components and routing of flow channels while following the design rules. After the flow channels have been routed, the channel lengths and therefore the routing latencies for the fluids that traverse these channels can now be calculated. Next, the given biochemical application is mapped onto this biochip architecture and the optimized schedule for its execution is generated. Based on the schedule, the control information (which valves to open and close at what time and for how long) can now be extracted. This is followed by the control layer routing and then the chip design is ready to be sent for fabrication.

Researchers have proposed placement algorithms [12, 14, 15, 16] for the flow layer, routing approaches for the flow layer [12, 17, 18], as well as integrated approaches for the placement and routing [12, 16]. Regarding the control layer, recent research has addressed the control channel routing [18, 19, 20]. For our control pin minimization strategy, we assume that we get as input an initial flow layer "netlist", i.e., the components and their interconnections, and that we know the channel delays, i.e., the flow channels have been routed. This is needed to determine the schedule for the operations in the biochemical application. However, once we perform the trade-off between off- and on-chip control in step 3, we will need to add the on-chip control to the biochip layout. We do this by using the flow-layer physical tools we have developed [12, 16]. Also, only after performing the valve minimization in step 4, we will know how the on-chip microvalves have to be connected to the output ports (and which microvalves share can share the control signals). We use our proposed physical synthesis tools for the routing of on-chip control channels [18, 20] to determine the physical layout of the control layer.

A. Control Synthesis

The control logic η is generated by fetching the control layer model of the biochip flow paths and components (part of the biochip architecture model), and utilizing them to translate the schedule into the valve activation sequence. At every time step of the schedule (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. 4a. Operation O_3 is bound to $Heater_1$, so we fetch the control layer model for $Heater_1$ from the table 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 (Fig. 4b). Similarly the control layer models for the flow paths F_{0-2} and F_{6-2} are fetched from the component library 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.

For our example, we need 89 control pins to control the 89 valves on the chip.

B. Valve Minimization

The pin count minimization problem has previously been reduced to a graph coloring problem (GCP) [22]. In GCP, the nodes in the coloring graph need to be colored using minimum number of colors, in such a way that no two adjacent nodes have the same color. The graph G is created by considering each valve as a separate node in the graph. An edge 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.

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 Fig. 4b 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 G by considering each valve in Fig. 4b as a separate node in the graph (redundant valves are not considered). An edge 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 Fig. 4b, 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 Fig. 4b has 83 nodes (total valves were 89, 6 were found to be redundant and were removed) and 1312 edges.

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

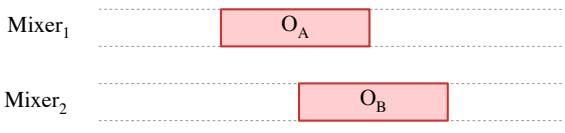
Considering the complexity of the problem, different metaheuristic techniques have also been used extensively for finding good graph coloring solutions, especially when there are a large number of nodes. We use a Tabu Search-based optimization scheme in order to perform the pin count minimization [13].

C. *Pin Count-Aware Compilation*

For our example, we need to control 89 valves. We are interested to reduce the number of needed control pins. For example, the MI chip has only 30 control pins. The maximum number theoretically available is limited by the size of the chip (each pin takes space) and the number of off-chip solenoid valves available.

Such control pin minimization approaches are applied after the schedule has been generated. However, the scheduling step offers a greater opportunity to reduce the number of control pins, e.g., by attempting to synchronize operations, allowing multiple valves to share the same control.

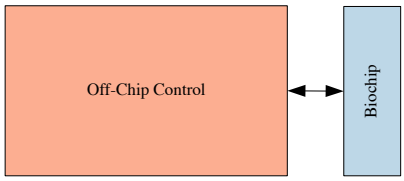
We propose a new scheduling technique, which produces a schedule such that the number of edges in the coloring graph G is minimized and the application deadline is satisfied. By minimizing the number of edges in the graph G during scheduling, we are able to significantly reduce the number of control pins compared to the related work.



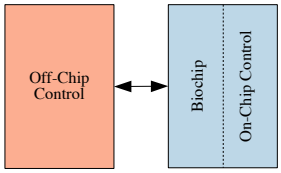
(a) No valve sharing: 18 control pins are needed to run the 2 mixers

(b) Overlapped operation: 9 control pins are needed to run the 2 mixers

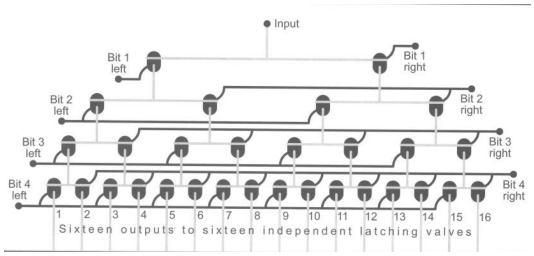
Fig. 5. Delaying operations to maximize overlap



(a) Expensive, bulky and power hungry on-chip control



(b) Part of the off-chip control has been moved on-chip



(c) 8:16 on-chip muliplexer for on-chip control [23]

Fig. 6. Trade-off between off- and on-chip control

The proposed technique is based on a List Scheduling (LS) heuristic [21]. LS heuristics use a sorted priority list, L_{ready}, containing the operations ready to be scheduled. An operation O_i is ready if all the predecessor operations have finished executing and all the incoming fluids are received. We use the "urgency" priority function [21] to sort L_{ready}. However, compared to the original LS, we do not immediately schedule a ready operation O_i. Let us assume that we would schedule O_i at time t_i. Instead, before scheduling O_i, we incrementally delay O_i with on time step, i.e., $t_i + 1$, $t_i + 2$, etc., up to $ALAP_i$, where $ALAP_i$ is the as-late-as-possible start of O_i such that the deadline D of the biochemical application is not exceeded. The value of $ALAP_i$ for each operation is determined by performing ALAP scheduling [21] on the application graph. For each increment $t_i + j$, we create a new coloring graph G, as discussed in Section IV.B. We record that increment j, which corresponds to the minimum number of edges in G. We then schedule O_i at time step $t_i + j$, with j as determined earlier.

Let us consider the example in Fig. 5, where we have already scheduled operation O_B on $Mixer_2$ as depicted in Fig. 5a, and we need to decide how to schedule operation O_A on $Mixer_1$. Let us assume that $Mixer_1$ is controlled by 9 valves, v_2 to v_{10}, and $Mixer_2$ by valves v_{28} to v_{36}, see Fig. 2a. If we schedule operation O_A at the earliest time when it is ready for execution, such as in Fig. 5a, we will need 18 control signals to operate the two mixers. However, if we delay the execution of O_A to the time depicted in Fig. 5b, we overlap the execution of the two mixers, such that their valves operate synchronously, and thus can share the control signals. In this case, only 9 control signals are needed to operate the two mixers.

Please note that this is a heuristic algorithm, i.e., it is not

guaranteed to lead to the minimum number of control pins. We are using the number of edges in *G* as a "proxy cost function" instead of actually finding out the number of "colors" in the graph *G*, which would be too time consuming to do within LS. However, for our example, we are able to reduce the number of control pins from 89 (which cannot be implemented on the MI chip) to a final number of 12 (after all the steps in the strategy), less than the 30 available pins, which shows that the heuristic is able to reduce the number of control pins during the scheduling step.

IV. Conclusions and Future Work

In this paper we have addressed flow-based biochips where the building block is a microvalve, which is used to build more complex components. We have proposed a general strategy to minimize the number of control pins needed to run a biochemical application on a given biochip architecture. The strategy consists of 5 steps, (1) compilation, (2) control synthesis, (3) on/off-chip control trade-off, (4) valve minimization and (5) physical synthesis. The focus of the paper was on the compilation step, which aims at minimizing the number of signals needed to control the microvalves, by maximizing the signal sharing among the microvalves. We have proposed a List Scheduling-based heuristic for the pin count-aware compilation (step 1), and we have discussed the existing solutions for steps 2 to 5. The proposed methodology was validated on a case study implemented using a microfluidic control system from MI. As future work, it would be interesting to propose a better solution to the compilation task, which uses as cost function the number of control pins, and not the number of edges in the control graph. This could be achieved also by integrating steps (1) compilation and (4) valve minimization into a single optimization problem. In addition, it would be interesting to investigate solutions to the on/off-chip control trade-off, by deciding automatically on the part of the off-chip control that should be moved on-chip and synthesizing the required on-chip circuitry and physical layout.

References

[1] I. E. Araci and S. R. Quake, "Microfluidic very large scale integration (mvlsi) with integrated micromechanical valves," *Lab Chip*, vol. 12, pp. 2830–2806, 2012.

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

[3] 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.

[4] K. Chakrabarty and J. Zeng. "Design automation for microfluidics-based biochips". Journal on Emerging Technologies in Computing Systems, 1(3): 186–223, 2005.

[5] http://research.microsoft.com/en-us/um/india/projects/biocoder/

[6] http://microfluidicinnovations.com

[7] N. Amin, "Computer-Aided Design for Multilayer Microfluidic Chips", master's thesis, Massachusetts Institute of Ttechnology, December 2008.

[8] 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., 39:1153–1182, 2010

[9] W. H. Grover, R. H. C. Ivester, E. C. Jensen, R. A. Mathies. "Development and multiplexed control of latching pneumatic valves using microfluidic logical structures", in *Lab on a Chip* 6, no. 5 623-631, 2006.

[10] T. V. Nguyen, S. Ahrar, P. N. Duncan, E. E. Hui. "Microfluidic finite state machine for autonomous control of integrated fluid networks", in *Proc. of Micro Total Analysis Systems*, pp. 741-743. 2011.

[11] P. N. Duncan, T. V. Nguyen, E. E. Hui. "Pneumatic oscillator circuits for timing and control of integrated microfluidics." *Proceedings of the National Academy of Sciences* 110, no. 45, 18104-18109, 2013.

[12] W. H. Minhass, P. Pop, J. Madsen, F. S. Blaga. "Architectural Synthesis of Flow-Based Microfluidic Large-Scale Integration Biochips", in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. Association for Computing Machinery, pp. 181– 190, 2012.

[13] W. H. Minhass, P. Pop, J. Madsen, T.-Y. Ho. "Control Synthesis for the Flow- Based Microfluidic Large-Scale Integration Biochips", in *Asia and South Pacific Design Automation Conference*, pp. 205–212, 2013.

[14] J. McDaniel, B. Parker, and P. Brisk, "Simulated Annealing-based Placement for Microfluidic Large Scale Integration (mLSI) Chips," in *Proceedings of the International Conference on Very Large Scale Integration* (VLSI-SoC), pp. 213–218, 2014.

[15] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho, "A top-down synthesis methodology for flow-based microfluidic biochips considering valve- switching minimization," in *Proceedings of the 2013 ACM international symposium on International symposium on physical design*, pp. 123–129, 2013.

[16] M. Raagaard, "Placement algorithm for flow-based microfluidic biochips," B.Sc. Thesis, Technical University of Denmark, 2014.

[17] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. Lee, and T.-Y. Ho, "An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips," in *Proceedings of the Design Automation Conference*, pp. 1–6, 2014.

[18] M. S. Hørslev-Petersen, T. O. Risager, "Routing algorithms for flow- based microfluidic very large scale integration biochips," B.Sc. Thesis, Technical University of Denmark, 2013.

[19] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer optimization for flow-based mVLSI microfluidic biochips," in *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, p. 16, 2014.

[20] T. S. Rasmussen, "Routing algorithm for the control layer of flow-based biochips", B.Sc. Thesis, Technical University of Denmark, 2014.

[21] O. Sinnen, *Task Scheduling for Parallel Systems*, John Wiley & Sons, 2007.

[22] 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.

[23] W. H. Grover, R. A. Mathies, "Monolithic Membrane Valves and Pumps", chapter in Lab-on-a-Chip Technology (Editors: Keith E. Herold and Avraham Rasooly), Caister Academic Press, 2009.