Chapter 2
Design Methodology for Flow-Based Microfluidic Biochips

Abstract This chapter presents an overview of the mVLSI biochips design and programming methodologies, highlighting the main tasks that have to be performed. The purpose is to explain how the methods presented in this book are used within a methodology and to define and illustrate the main tasks. We highlight the difference between "physical design and testing," which is concerned with designing a biochip, and "programming and control," which considers that the biochip is given, and addresses the mapping of a biochemical application on the given architecture. Programming and control is covered by Part II, whereas physical design and testing are covered in Part III. This chapter also presents the related work for the design tasks introduced.

In academia, a significant amount of work has been carried out on the individual microfluidic components [20, 22]. The manufacturing technology, soft lithography, used for the flow-based biochips has advanced faster than Moore's law [13]. Although biochips are becoming more complex everyday, computer-aided design (CAD) tools for these chips are still in their infancy. Most CAD research has been focused on device-level physical modeling of components [19, 37].

Significant work on top-down synthesis methodologies for droplet-based biochips has been proposed [6, 32]. In these chips, the liquid is manipulated as discrete droplets on an electrode array. The synthesis process, starting from a given biochemical application and a droplet-based biochip architecture model, determines the resource allocation, binding, scheduling, and placement of the application operations. However, the architecture of the droplet-based chips differs significantly from the flow-based chips. In the flow-based biochips, components of different types (e.g., mixers, heaters) are physically designed on the chip and connected to each other using microfluidic channels. Once fabricated, the number and type of the components, their placement scheme on the chip, and the routing interconnections cannot be modified [25]. Droplet-based biochips (as discussed in Sect. 1.1), however, use the idea of virtual components and are reconfigurable. Because of the architectural differences, the models and techniques proposed for the digital biochips are not applicable to their flow-based counterparts.

The industry has gotten around the limited CAD tools problem by limiting the number of chips that they design and using them for multiple applications (Fluidigm Corporation has only 4 chip designs [10]). The soft lithography-based fabrication

P. Pop et al., *Microfluidic Very Large Scale Integration (VLSI)*,
DOI 10.1007/978-3-319-29599-2_2

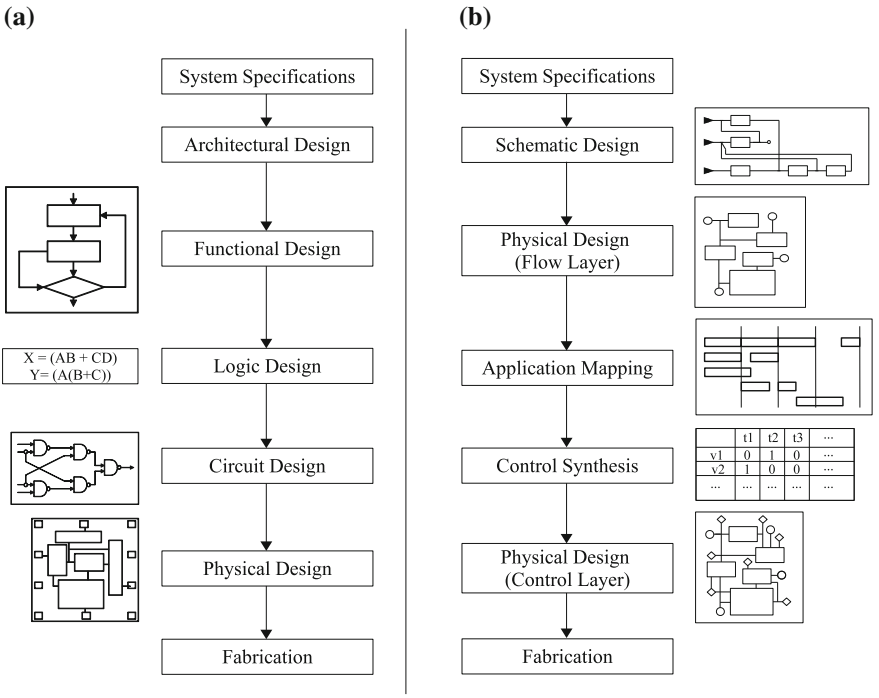

Fig. 2.1 VLSI versus mVLSI design cycles

process is, however, cheap and has a fast turnaround time [25]. Section 1.2.2 has motivated the need for CAD support during the physical design and testing of mVLSI biochips. Figure 2.1a shows the simplified design flow for the microelectronics VLSI (Very Large Scale Integration). Each design task is presented as a box. For example, VLSI design [36] starts from the system specifications which is followed by the architectural and functional design tasks. Then, the logic design of the chip is done, after which the circuit and physical design tasks are carried out. The chip is then sent for fabrication. The VLSI design process is described here in a linear manner for simplicity. In reality, each design task is followed by a verification step. Based on the verification result, there are many iterations back and forth in the design flow in order to meet the system specifications. Motivated by the similarity between VLSI and mVLSI, we propose the mVLSI design flow shown in Fig. 2.1b.

Recent research on mVLSI design methods has started to address design tasks in this design flow. We will refer to these results when describing the design tasks. A good overview of the recent developments in mVLSI is presented in [4]. Section 2.2 explains the design tasks involved in the biochip physical synthesis and testing, highlighting also the related work.

Section 1.2.3 highlights the need for programming languages and tools, as well as cheap and reliable control solutions. To implement a protocol (the "program" running on the biochip), the current practice is to manually convert each protocol operation into control signals for individual components. This is similar to how the first computers were programmed, toggling individually switches. This requires expertise in microfluidics and in hardware/software engineering (besides biochemistry, for the protocols), it is very time-consuming and error-prone, and makes it impossible to optimize the use of expensive agents and hard-to-obtain samples. Computers are nowadays programmed using high-level languages (HLLs). The objective of HLLs used in computers, such as Java or C++, is to increase the productivity and quality, and to tackle complexity. The typical flow is: the programmer writes the desired functionality in an HLL; the HLL is then translated by a compiler (which also does optimizations) into assembly code; the assembly code is then translated into machine code (that the computer understands) using an assembler; throughout, the programmer uses simulators and debuggers to understand and test the program. Motivated by the area of compiler technology, we have proposed in Fig. 2.2 a programming and control tool flow. The vision is to have an HLL for biochips where a user could write "MIX a:b IN RATIOS 1:2 for 30 s" and have a compiler and the associated software tools take care of the rest. This would be similar to how computers are programmed today. Section 2.3 presents the design tasks involved in the programming and control of mVLSI biochips, including a presentation of the related work.

2.1 Modeling and Simulation

The mVLSI design and programming flows are supported by models. All the tasks, related to physical design and testing, and related to programming and control, rely on the use of models. Modeling provides an abstraction of the physical implementation and illuminates the behavior and relationships of key issues. In this context, simulation is the modeling of dynamic behavior.

Regarding programming, languages such as BioStream and Aqua, see Sect. 1.2.3, are a high-abstraction level model of the behavior of the biochemical application. At a lower abstraction level, researchers have proposed a graph model for biochemical applications, where each node is an operation, and edges capture the fluid transport, see Fig. 2.3a. The mentioned biochemical application models are presented in Chap. 4. A simulation framework is presented in Sect. 6.5.

Regarding physical design, we have proposed a topology graph-based system-level model of a biochip architecture, which is independent of the underlying biochip implementation technology (see Fig. 2.3b) [28]. The topology graph is also called a *netlist*, and it captures the components in the architecture and their interconnections, at a high-level of abstraction. However, to fabricate a biochip, a more detailed model is required. We have also proposed a *component model*, which captures the details of

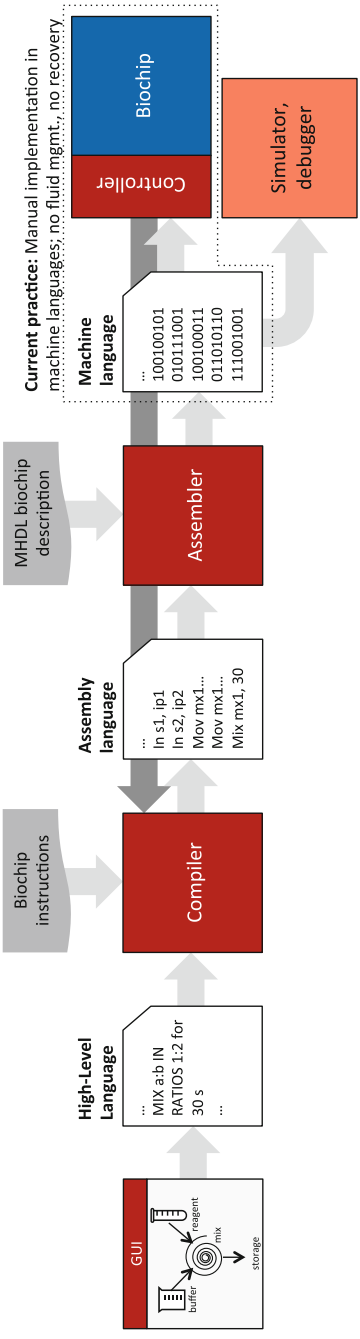

Fig. 2.2 Programming and control: design tasks and tools

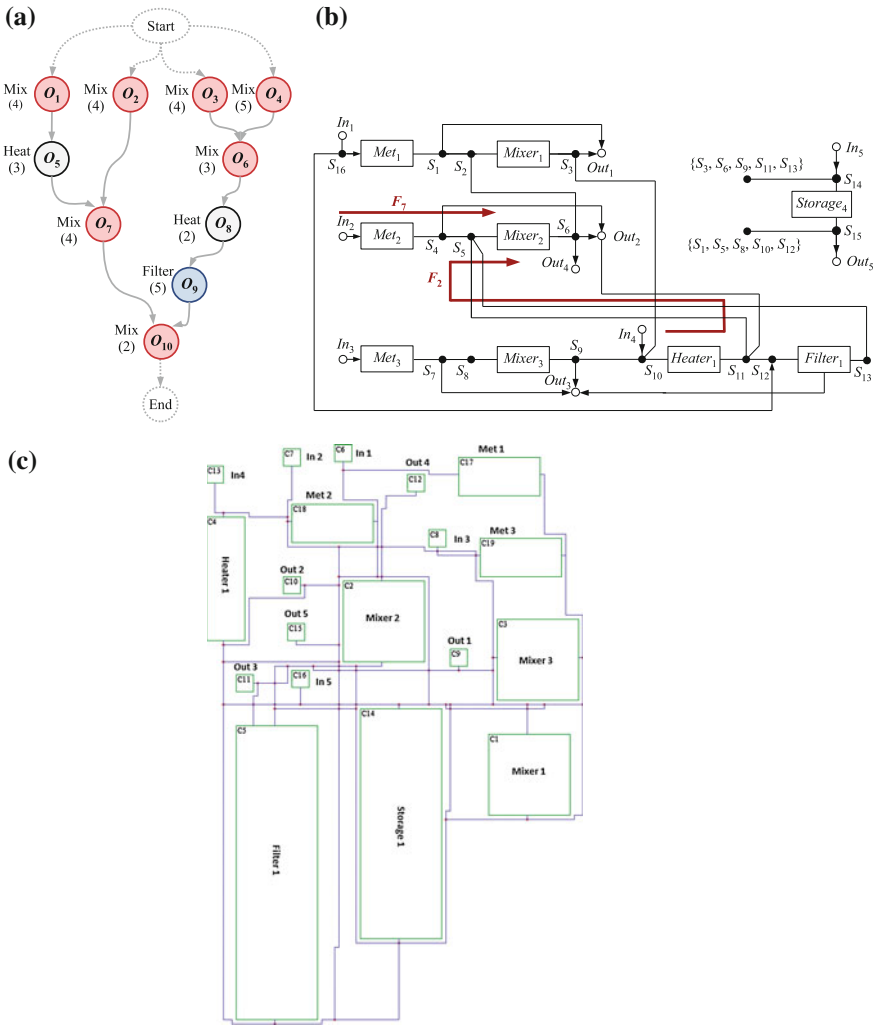

Fig. 2.3 Biochip application and architecture example. **a** Application model. **b** Biochip architecture model. **c** Placement and routing example

the components. The physical design tasks rely on a *grid model* to decide on the placement of components and the routing of interconnections. For the testing and design for dependability tasks, we use a *fault model* proposed in the literature [17]. Researchers have suggested a "Microfluidic Hardware Description Language" (MHDL) [24] to capture the biochip architecture throughout the various design tasks. The biochip architecture models we use in the book are presented in Chap. 3.

2.2 Physical Design and Testing

Allocation and schematic design. Given the system specifications (e.g., application requirements, chip area), the mVLSI design flow starts with the allocation and schematic design of the required biochip. In this step, the microfluidic components required for implementing a given biochemical application are allocated from a component library, while taking into account the imposed resource constraints. Next, based on the given application, a chip schematic is designed and the netlist is generated. For example, to implement the biochemical application from Fig. 2.3a under the constraints given in Table 2.1 columns 1 and 2, we could use an allocation such as the one captured by the last two columns in Table 2.1. The schematic design corresponding to such an application and allocation is presented in Fig. 2.3b. Note that the storage units are needed in order to save the output of a component so that it can be used at a later stage. The flow path set is also generated in this step. A *flow path* is the path starting from the point of fluid sample origin and ending at the fluid sample destination point, e.g., $Heater_1$ to $Mixer_2$ in Fig. 2.3b. Source-sink paths associated with each flow path are also defined, e.g., for the flow path $Heater_1$ to $Mixer_2$ in Fig. 2.3b, the source-sink path is (In_4, S_{10}, $Heater_1$, S_{11}, S_5, $Mixer_2$, S_6, Out_2). Routing constraints are also extracted at this stage. Two flow paths, whose corresponding source-sink paths have a common vertex are mutually exclusive and need to be listed as routing constraints, e.g., F_7 and F_2 in Fig. 2.3b are mutually exclusive since they share common vertices (e.g., S_5) in their source-sink paths.

Placement and routing. The allocation and schematic design 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. In this step, the allocated components are placed on a chip layout area and the interconnections between components are routed as channels on the chip such that the application completion time is minimized. The placement and routing phases are governed by design rules imposed by the fabrication process. During placement, the components are treated as fixed size blocks, represented by rectangles, each having a fixed length and width. The placement is done in such a way that all design rules are satisfied and no two components overlap on the chip. For mVLSI-based biochips, the placement and routing phases

Table 2.1 Allocated Components

Function	Constraints	Allocated units	Notations
Input port	5	5	In_1 ... In_5
Output port	5	5	Out_1 ... Out_5
Mixer	3	3	$Mixer_1$ ... $Mixer_3$
Heater	2	1	$Heater_1$
Filter	1	1	$Filter_1$
Metering units	3	3	Met_1 ... Met_3
Storage units	4	4	$Storage_x$

can be divided into two stages, one for each logical layer in the chip: the flow layer and the control layer.

Flow Layer. This stage involves determining the placement of microfluidic components and the fluidic inlet and outlet ports on the chip layout area, and then routing the interconnecting channels as microfluidic flow channels. In VLSI chips, the intersection of nets is considered a short-circuit and is thus not permitted. However, net intersection is possible in the biochip flow layer. A switch, composed of four valves controlling the channel intersection, is placed at the location of the intersection so that both channels can be used, at different points in time, without unintended fluid mixing. Considering that only one layer is available for routing all flow channel nets, the possibility of net intersection helps in achieving 100 % routability. However, net intersections cause routing constraints, resulting in longer application completion times. Figure 2.3c shows a placement and routing scheme for the flow layer of the biochip architecture shown in Fig. 2.3b. Researchers have proposed placement algorithms [23, 28, 34, 41] for the flow layer, routing approaches for the flow layer [14, 21, 28], as well as integrated approaches for the placement and routing [28, 34]. After the flow channels have been routed, the channel lengths and therefore the routing latencies for the fluids that traverse these channels can be calculated.

Control Layer. In this stage, the placement of the control valves and the control ports is decided, and then the valves are connected to the control ports through control channel routing. The positions of the valves that are used inside a microfluidic component can be obtained directly from the component library. The positions of the valves that need to be placed on the flow channels are inferred from the flow routing information (e.g., valves need to be placed at all flow channel intersections). Contrary to the flow channels, control channels are not allowed to intersect. After the placement is complete, the next step is to connect the valves to the ports using control channels. The control channels can be routed over/under any flow channel/component without forming a valve. The crossing of the control channel over a flow channel forms a valve only if the control channel has a larger width. The flow path channel lengths (used to calculate the routing latencies) and any additional routing constraints (imposed because of net intersections in the flow layer) can now be extracted from the layout and captured in the biochip architecture model. Researchers have started to address control channel routing [14, 16].

On-chip control synthesis. 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. The number of controls is limited. For example, an mVLSI biochip control system from Microfluidics Innovation, LLC, has 36 control pins, interfaced to the biochip through a manifold. Although recent research has proposed top-down physical synthesis methods and tools, and programming languages and compilation techniques to automatically derive the control signals for the valve actuations (see Sect. 2.3), 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.

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 pneumatic transistors, multiplexers, memory latches, logic gates, shift-registers, adders, and even clocks and finite-state machines [8, 12, 30]. The vision is to move all the off-chip control on-chip, removing the dependence on external control completely.

Let us discuss on-chip multiplexers in more detail. Microfluidic multiplexers [25] 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 $2\log_2 N$ control pins for controlling N such flow channels [1], e.g., 3 pins will be sufficient to control 8 such flow channels as shown in Fig. 2.4. Valves are created only where a wide control channel (red) intersects with a flow channel (blue). Narrow intersections do not create valves. In Fig. 2.4, control channels with status 1 have the associated valves closed and the ones with status 0 have the associated valves open. For the case shown, the valves are activated such that flow channel 3 is open and all others are closed.

In order to perform the multiplexer-based 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 [1]. Once the flows have been specified, 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 overconstrained specifications, resulting in inefficient

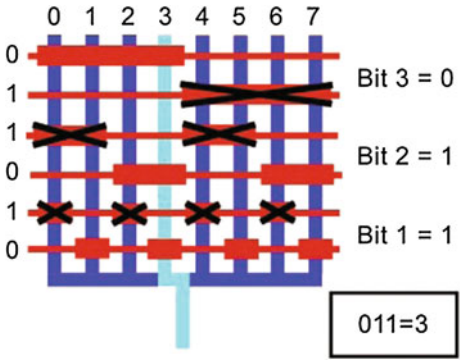

Fig. 2.4 Microfluidic multiplexer; From Thorsen, T., Maerkl, S.J., Quake, S.R.: Microfluidic large-scale integration. Science 298(5593), 580–584 (2002). Reprinted with permission from AAAS

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.

Testing and design for dependability. Physical defects can be introduced while manufacturing mVLSI biochips [17]. Figure 1.3 shows an example biochip schematic and a block defect that manifested in the flow channel during the fabrication. More such defects are presented in Fig. 11.1. These fabrication defects can be costly because of the reduced manufacturing yield, the need to redo lengthy experiments, using expensive reagents and often hard-to-obtain samples, and can be safety-critical, e.g., in the case of a cancer misdiagnosis. Additionally, with increase in complexity of these biochips (as mentioned, commercial mVLSI biochips are available which use more than 25,000 microvalves), the fabrication constraints become tighter, thereby increasing the chance of defect occurrence.

As mentioned, the current practice is to manually screen biochips for defects under expensive microscopes, using cameras, scanning, and vision analysis software. This technique is labor-intensive, has poor fault-coverage, and can lead to over-testing. Recent work has addressed the automated testing of mVLSI biochips by converting their structure into a logic circuit model composed of Boolean gates, using air pressure as test signals [5, 17, 33], and using standard test pattern generation techniques. In addition, the problem of fault tolerance in mVLSI biochips has also been addressed recently [9, 31].

2.3 Programming and Control

Compiling high-level languages. Figure 2.2 presents our proposed programming and control tool flow. The biochemist end-user can specify the biochemical protocol using a *graphical user interface* (GUI). The visual representation has a textual description equivalent, which is a *high-level protocol language*, or HLL. Researchers have proposed HLLs for describing biochemical protocols, such as the Aqua language [3] and BioStream [40]. A *controller*, attached to the programmable biochip, controls the valves to implement the biochemical application captured by the HLL. The opening and closing of valves is done according to a valve actuation sequence, the *machine language* in Fig. 2.2, which is what the controller understands. To run the biochemical application, its description in the HLL has to be translated to this machine language. This is the task of the *compiler* and *assembler*. The compiler has to know the supported biochip instructions, in order to produce an *assembly language*, which is an intermediate language specific to the particular biochip (at a higher abstraction level than the machine language, but lower abstraction than the HLL). The assembler has to know the details of the biochip implementation, which could, for example, be captured using an MDHL [24].

The assembler and compiler work on an internal representation model for the biochemical application. Chapter 4 presents both an HLL considered and the application sequencing graph model we use in the book as an internal representation. The first step in the compilation process is to generate this application graph from an HLL.

This issue has been addressed in [2, 18, 40] for continuous-flow biochips and in [11] for droplet-based biochips. During the generation of the application graph model we have to solve the "mixing problem," i.e., how to generate a mixing tree to perform an arbitrary mixing ratio, as specified in the high-level language, considering 1:1 mixing operations, typically available on the biochips [40].

Binding, scheduling, and fluid routing. The compilation task has to solve next the binding, scheduling, and fluid routing problems, which are collectively called the "application mapping" problem. Currently, researchers manually map the applications to the valves of the chip using a custom interface (analogous to exposure of gate-level details) [39]. The manual process is quite tedious and needs to be repeated every time a change is made either to the chip architecture or the biochemical application. For larger chips and applications, the process can easily result in inefficient application mappings. Researchers have proposed significant work on application mapping for droplet-based biochips [6]. However, due to the differences between the two technologies, as discussed earlier, these techniques are not applicable to the flow-based chips.

The biochemical application has to be mapped onto the synthesized architecture such that the application completion time is minimized and the dependency, resource, and routing constraints are satisfied. The binding for the operations can be determined when generating the schematic, or can be modified in the scheduling step. The binding of the edges and the scheduling for both the operations and the edges is generated now. Figure 2.5b shows the schedule for the case when the application in Fig. 2.3a is scheduled on the architecture in Fig. 2.3b. The schedule is represented as a Gantt chart, where we represent the operations and fluid routing phases as rectangles, with their lengths corresponding to their execution duration.

Researchers have started to propose approaches to the application mapping for flow-based biochips [7, 26, 27, 41]. In the context of application mapping, fluid routing has a significant impact on the quality of the solution. However, initial work on scheduling has ignored fluid routing [27]. In the case of Stanford valves, to be able to transport a fluid inside a flow channel, we need to apply pressure through

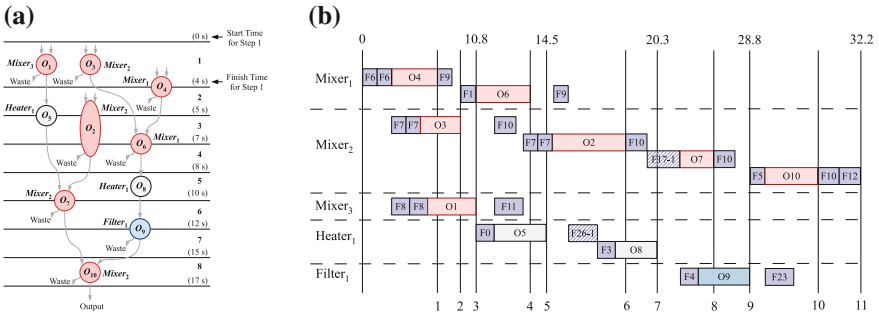

Fig. 2.5 Illustrative example. **a** Allocation Example for Fig. 2.3a. **b** Schedule

an input port (or via an on-chip micropump) connected to the channel. To allow fluid movement, the channel also needs to be connected to an output port. This complicates the fluid routing problem, because for every fluid transport operation we need to identify corresponding input and output ports. To simplify this routing task, all the research so far [7, 26] has assumed that each on-chip component is connected to implicit input and output ports, which is unrealistic.

In addition, some biochemical applications contain fluids that may adsorb on the substrate on which they are transported. Consequently, the purity of other fluids on the biochip may be affected by the contaminated areas, and this could lead to an erroneous outcome. The solution is to use strategies for contamination avoidance and "washing" of contaminated regions. The problem of washing has been addressed *separately* from the scheduling and fluid routing problems [15] in which they wash the chip after the assay has completed, preparing it for a second experiment.

Control synthesis. Based on the schedule, the control information (which valves to open and close at what time and for how long) can now be extracted. Optimization schemes can be used to minimize the chip pin-count in the control layer, reducing the macro-assembly around the chip. This is followed by the control layer routing and then the chip design is ready to be sent for fabrication. Recent research has addressed the control-pin minimization problem both before [29] and during [35] the scheduling task. The control pin-count minimization problem has been reduced to the graph-coloring problem (which is NP-hard) [1], and a solution was presented in [29].

The pressure signals to the microvalves are delivered via a control box. Professor Quake's group at Stanford University [38] has developed a USB-based valve control system to drive the valves in the chip, from a computer using LabView or Matlab. Microfluidic Innovations has a control box which receives the HLL Aqua as input [3]. A cheap and scalable solution based on an Arduino control board has also been developed [42].

References

1. Amin, N., Thies, W., Amarasinghe, S.: Computer-aided design for microfluidic chips based on multilayer soft lithography. In: Proceedings of the IEEE International Conference on Computer Design, pp. 2–9 (2009)
2. Amin, A.M., Thottethodi, M., Vijaykumar, T.N., Wereley, S., Jacobson, S.C.: Automatic volume management for programmable microfluidics. In: Proceedings of the 2008 ACM SIGPLAN Conference on Programming Language Design and Implementation (2008)
3. Amin, A.M., Thakur, R., Madren, S., Chuang, H.S., Thottethodi, M., Vijaykumar, T., Wereley, S.T., Jacobson, S.C.: Software-programmable continuous-flow multi-purpose lab-on-a-chip. Microfluid. nanofluid. **15**(5), 647–659 (2013)
4. Araci, I.E., Brisk, P.: Recent developments in microfluidic large scale integration. Curr. Opin. Biotechnol. **25**(60–68) (2014)
5. Araci, I., Pop, P., Chakrabarty, K.: Microfluidic very large-scale integration for biochips: Technology, testing and fault-tolerant design. In: 20th IEEE European Test Symposium, pp. 1–8 (2015)

6. Chakrabarty, K., Xu, T.: Digital Microfluidic Biochips: Design Automation and Optimization. CRC Press, Boca Raton (2010)
7. Dinh, T.A., Yamashita, S., Ho, T.Y., Hara-Azumi, Y.: A clique-based approach to find binding and scheduling result in flow-based microfluidic biochips. In: Proceedings of 18th Asia and South Pacific Design Automation Conference, pp. 199–204. IEEE (2013)
8. Duncan, P.N., Nguyen, T.V., Hui, E.E.: Pneumatic oscillator circuits for timing and control of integrated microfluidics. Proc. Natl. Acad. Sci. **110**(45), 18104–18109 (2013)
9. Eskesen, M.C., Pop, P., Potluri, S.: Architecture synthesis for cost-constrained fault-tolerant flow-based biochips. In: Design, Automation and Test in Europe Conference and Exhibition (2016, in press)
10. Fluidigm: Chips and Kits. http://www.fluidigm.com/chips-kits.html
11. Grissom, D., Brisk, P.: A high-performance online assay interpreter for digital microfluidic biochips. In: Proceedings of the Great Lakes symposium on VLSI, pp. 103–106 (2012)
12. Grover, W.H., Ivester, R.H., Jensen, E.C., Mathies, R.A.: Development and multiplexed control of latching pneumatic valves using microfluidic logical structures. Lab Chip **6**(5), 623–631 (2006)
13. Hong, J.W., Quake, S.R.: Integrated nanoliter systems. Nat. Biotechnol. **21**, 1179–1183 (2003)
14. Hørslev-Petersen, M.S., Risager, T.O.: Routing algorithms for flow-based microfluidic very large scale integration biochips. Master's thesis, Technical University of Denmark (2013)
15. Hu, K., Ho, T.Y., Chakrabarty, K.: Wash optimization for cross-contamination removal in flow-based microfluidic biochips. In: Proceedings of 18th Asia and South Pacific Design Automation Conference, pp. 244–249 (2014)
16. Hu, K., Dinh, T.A., Ho, T.Y., Chakrabarty, K.: 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. ACM (2014)
17. Hu, K., Yu, F., Ho, T.Y., Chakrabarty, K.: Testing of flow-based microfluidic biochips: fault modeling, test generation, and experimental demonstration. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **33**(10), 1463–1475 (2014)
18. Kaas-Olsen, M.: Synthesis of flow-based biochip architectures from high-level protocol languages. bachelorsthesis, Technical University of Denmark (2015)
19. Klammer, I., Buchenauer, A., Fassbender, H., Schlierf, R., Dura, G., Mokwa, W., Schnakenberg, U.: Numerical analysis and characterization of bionic valves for microfluidic PDMS-based systems. J. Micromech. Microeng. **17**(7), S122–S127 (2007)
20. Lim, Y.C., Kouzani, A.Z., Duan, W.: Lab-on-a-chip: a component view. J. Microsyst. Technol. **16**(12) (2010)
21. Lin, C.X., Liu, C.H., Chen, I.C., Lee, D., Ho, T.Y.: An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips. In: Proceedings of the Design Automation Conference, pp. 1–6. ACM (2014)
22. Mark, D., Haeberle, S., Roth, G., Stetten, F., Zengerle, R.: Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications. Chem. Soc. Rev. **39**, 1153–1182 (2010)
23. McDaniel, J., Parker, B., Brisk, P.: Simulated annealing-based placement for microfluidic large scale integration (mLSI) chips. In: Proceedings of the International Conference on Very Large Scale Integration, pp. 213–218 (2014)
24. McDaniel, J., Baez, A., Crites, B., Tammewar, A., Brisk, P.: Design and verification tools for continuous fluid flow-based microfluidic devices. In: Proceedings of 2013 18th Asia and South Pacific Design Automation Conference, pp. 219–224. IEEE (2013)
25. Melin, J., Quake, S.: Microfluidic large-scale integration: the evolution of design rules for biological automation. Ann. Rev. Biophys. Biomol. Struct. **36**, 213–231 (2007)
26. Minhass, W.H., Pop, P., Madsen, J.: System-level modeling and synthesis of flow-based microfluidic biochips. In: Proceedings of the International Conference on Compilers, Architectures and Synthesis of Embedded Systems (2011)
27. Minhass, W.H., Pop, P., Madsen, J.: Synthesis of Biochemical Applications on Flow-Based Microfluidic Biochips using Constraint Programming. In: Proceedings of the IEEE Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS, pp. 37–41 (2012)

28. Minhass, W.H., Pop, P., Madsen, J., Blaga, F.S.: Architectural synthesis of flow-based microfluidic large-scale integration biochips. In: Proceedings of the International Conference on Compilers, Architectures and Synthesis of Embedded Systems, pp. 181–190 (2012)
29. Minhass, W.H., Pop, P., Madsen, J., Ho, T.: Control synthesis for the flow-based microfluidic large-scale integration biochips. In: Proceedings of the Asia and South Pacific Design Automation Conference (2013)
30. Nguyen, T., Ahrar, S., Duncan, P., Hui, E.: Microfluidic finite state machine for autonomous control of integrated fluid networks. In: Proceedings of Micro Total Analysis Systems. pp. 741–743
31. Pop, P.: Flow-based biochips: Fault-tolerant design and error recovery. In: Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2015)
32. Pop, P., Maftei, E., Madsen, J.: Recent research and emerging challenges in the system-level design of digital microfluidic biochips. In: Proceedings of the IEEE International SOC Conference (2011)
33. Pop, P., Araci, I.E., Chakrabarty, K.: Continuous-flow biochips: technology, physical design methods and testing. IEEE Design Test **32**(6), 8–19 (2015)
34. Raagaard, M.: Placement algorithm for flow-based microfluidic biochips. B.Sc. thesis, Technical University of Denmark (2014)
35. Raagaard, M.L., Pop, P.: Pin count-aware biochemical application compilation for mVLSI biochips. In: Symposium on Design Test Integration and Packaging of MEMS/MOEMS, pp. 1–6 (2015)
36. Sait, S.M., Youssef, H.: VLSI Physical Design Automation: Theory and Practice. World Scientific Publishing Co., Pte. Ltd, NJ (1999)
37. Siegrist, J., Amasia, M., Singh, N., Banerjee, D., Madou, M.: Numerical modeling and experimental validation of uniform microchamber filling in centrifugal microfluidics. Lab Chip **10**, 876–886 (2010)
38. Stanford Microfluidic Foundry. http://www.stanford.edu/group/foundry/
39. Thies, W.B.: Programmable microfluidics. presented at Stanford University (2007)
40. Thies, W., Urbanski, J.P., Thorsen, T., Amarasinghe, S.: Abstraction layers for scalable microfluidic biocomputing. Natl. Comput. **7**(2), 255–275 (2008)
41. Tseng, K.H., You, S.C., Liou, J.Y., Ho, T.Y.: 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. ACM (2013)
42. Understrup, K.G.: Programming and control of flow-based microfluidic biochips. Master's thesis, Technical University of Denmark (2014)

http://www.springer.com/978-3-319-29597-8

Microfluidic Very Large Scale Integration (VLSI)
Modeling, Simulation, Testing, Compilation and Physical Synthesis

Pop, P.; Minhass, W.H.; Madsen, J.

2016, XV, 270 p. 148 illus., 101 illus. in color.,
Hardcover
ISBN: 978-3-319-29597-8