Cloud Columba: Accessible Design Automation Platform for Production and Inspiration

(Invited Paper)

Tsun-Ming Tseng[†], Mengchu Li[†], Yushen Zhang[†], Tsung-Yi Ho[⋆◊], and Ulf Schlichtmann[†]

{tsun-ming.tseng, mengchu.li, yushen.zhang, ulf.schlichtmann}@tum.de, tyho@cs.nthu.edu.tw

[†]Chair of Electronic Design Automation, Technical University of Munich, Arcisstraße 21, 80333 München, Germany

[⋆]Department of Computer Science, National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, 30013 Hsinchu, Taiwan

[◊]Institute for Advanced Study, Technical University of Munich, Lichtenbergstraße 2 a, 85748 Garching, Germany

***Abstract*—Design automation for continuous-flow microfluidic large-scale integration (mLSI) biochips has made remarkable progress over the past few years. Nowadays a biochip containing up to hundreds of components can be automatically synthesized within a few minutes. However, the current advanced design automation tools are mostly developed for research use, which focus essentially on the algorithmic performance but overlook the accessibility. Therefore, we have started the Cloud Columba project since 2017 to provide users from different backgrounds with easy access to the state-of-the-art design automation approaches. Without being limited by the computing power of their end devices, users just need to formulate their design requests in a high abstraction level, based on which the cloud server will automatically synthesize a customized manufacturing-ready biochip design, which can be viewed and stored using simply a web browser. With the computer-synthesized designs, Cloud Columba supports application developers to explore a wider range of possibilities, and algorithm developers to validate and improve their ideas based on a practical foundation.**

I. Introduction

Continuous-flow microfluidic large-scale integration (mLSI) is a lab-on-a-chip technology that enables efficient and precise control of fluids in a miniaturized chip [1], [2]. With the integration of hundreds to thousands of microchannels and micromechanical valves, mLSI chips support numerous biochemical and biological applications such as polymerase chain reaction (PCR) [3], cell culture and monitoring [4], single-cell mRNA isolation [5], and chromatin immunoprecipitation (ChIP) [6], [7], etc.

Unlike large-scale integration of electronic circuits which employs a top-down design approach with clear rules and a mature toolkit, the development of mLSI is still in its infancy. Most mLSI chips have been designed manually following the designers' intuition, which results in a time-consuming and error-prone procedure.

Design automation for mLSI thus arose to alleviate design difficulty and to enhance design quality. With a decade of effort, current design automation approaches have addressed a wide scope of design problems including resource prediction and utilization [8], [9], scheduling and fluid routing [10], storage and caching [11], [12], sample dilution and mixing [13], fault tolerance [14], reliability [15], [16], and security [17], etc. However, due to the lack of accessible mLSI physical design tools, most of the research stays on a high abstraction level, where the microfluidic components are treated as symbols and their geometric features are mostly omitted or simplified. This abstraction leads to inaccuracy in the estimation of the algorithmic performance and becomes an obstacle in the development process.

Actually, automatic physical synthesis of mLSI chips has been actively studied in recent years [18]–[20]. In particular, the state-of-the-art physical synthesis tools *Columba 2.0* [21] and *Columba S* [22] have demonstrated their ability of synthesizing manufacturing-ready mLSI designs within a few minutes or seconds. However, these tools were not easily accessible for both design automation researchers and bio-application developers, due to the complexity in algorithmic implementation and the demand for software environment and computing power.

To provide people from different backgrounds with easy access to customized and manufacturable mLSI designs, we propose *Cloud Columba*, an online-platform for automatic physical synthesis of mLSI designs. As shown in Figure 1, users of Cloud Columba just need an internet-connected web browser to specify their high-level design requests, and the cloud server will perform the state-of-the-art algorithms to return an mLSI design that can be directly exported for fabrication. With Cloud Columba, the whole design procedure

ICCAD 2019, DOI: 10.1109/ICCAD45719.2019.8942104
http://ieeexplore.ieee.org/document/8942104/

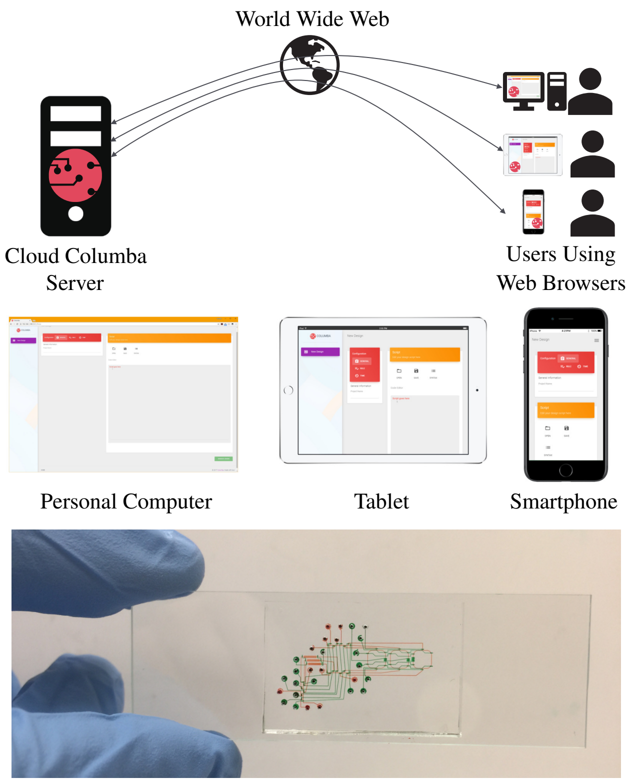

Figure 1: Cloud Columba is easily accessible with a web browser on various devices since all computational tasks are performed on the cloud server. It outputs a manufacturing-ready mLSI design that is customized based on user requests.

is shortened from days or weeks to a few minutes, and no specialized interdisciplinary knowledge or computationally powerful device is required.

The paper is organized as follows: Section II introduces the modular mLSI design concept; Section III describes the synthesis flow of Cloud Columba with an example; Section IV discusses the services that will be integrated into Cloud Columba in the near future; and Section V concludes this paper.

II. Cloud Columba: Modular mLSI Design

Cloud Columba aims to provide a modular interface for users to design their mLSI chips. Specifically, users just need to describe their requested devices and design their logic connections, and the cumbersome multi-layer physical implementation of valves and channels will be automatically performed with Columba 2.0/S by the cloud server. To this end, there are two prerequisites:

1) A descriptive method for users to specify their requested devices in a precise and consistent manner; and
2) An efficient place & route method that can easily adapt to various layout constraints.

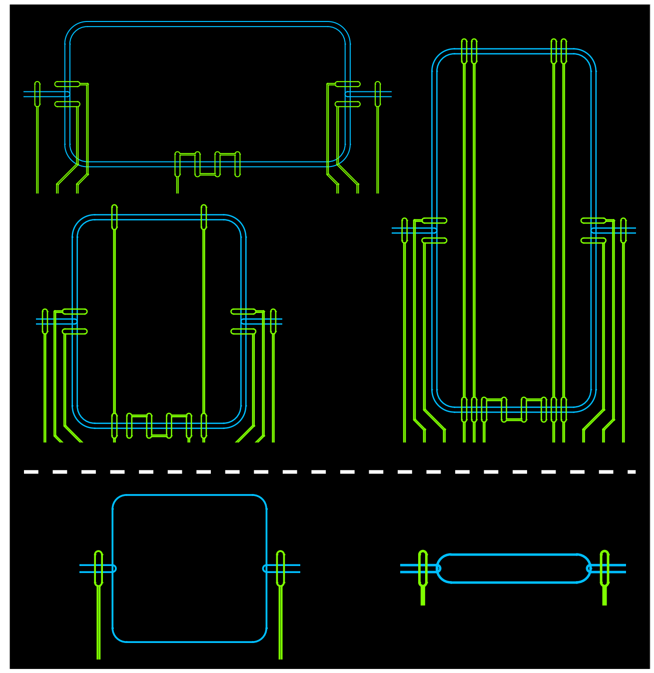

Figure 2: Devices synthesized by Cloud Columba. Flow channels are colored blue and control channels and valves are colored green. Above: rings of different dimensions embedded with peristaltic pumps, sieve valves, and separation valves. Below: chambers of different dimensions.

A. Component-Oriented General Device

To meet the first prerequisite, Cloud Columba adopts the component-oriented *general device* concept proposed in [9]. Specifically, it classifies on-chip microfluidic components into two categories: *containers* and *accessories*.

- Containers are microfluidic components that occupy exclusive chip area. Cloud Columba currently supports two types of containers: *chambers* and *rings*. A chamber consists of a rectangular channel segment, and a ring consists of a looped channel segment.
- Accessories are microfluidic components that can be embedded into containers and thus does not occupy exclusive chip area. Cloud Columba currently supports three types of accessories: *peristaltic pumps* [6], *sieve valves* [23], and *separation valves* [24].

Based on this classification, a *general device* is defined as a container embedded with zero to multiple accessories. In particular, the container defines the geometric features of a device, and the accessories define the functional add-ons that the device support. For example, a standalone chamber can serve as the device for metering, cell-culture, and dilution; and a ring embedded with peristaltic pumps can serve as the device for rotary mixing, etc. Whenever new microfluidic

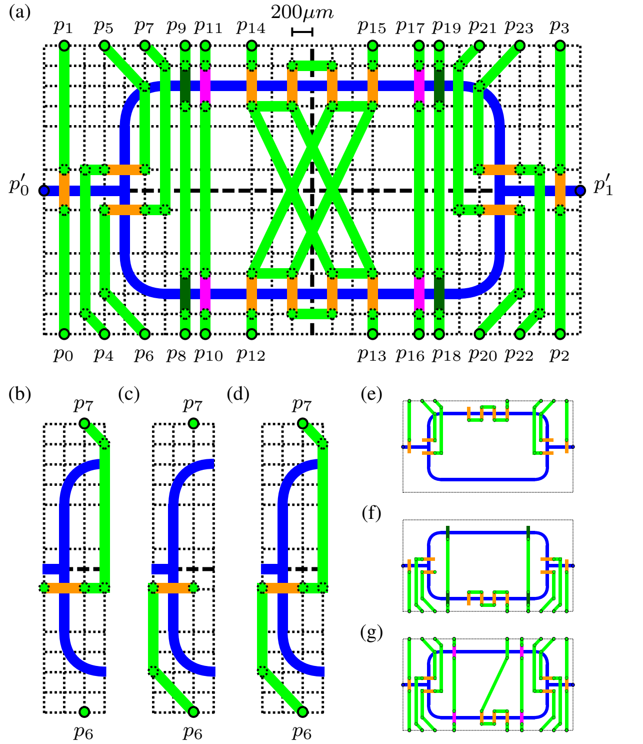

Figure 3: (a) Module model for a ring container, which can be embedded with peristaltic pumps, sieve valves, or separation valves. (b) Selecting pin p_7 allows the valve to be accessed from above. (c) Selecting pin p_6 allows the valve to be accessed from below. (d) Selecting both p_7 and p_6 allows the valve to be accessed from both directions. (e)(f)(g) Three possible implementations of the ring in the final layout.

components are invented, the container- and accessory-library can be extended to include the new inventions.

Users of Cloud Columba can thus describe their requested devices precisely and consistently by identifying their container categories together with their lengths, widths, and heights, and specifying the accessories that they want to integrate. Figure 2 demonstrates some exemplary devices of various dimensions and embedded with various accessories synthesized by Cloud Columba.

B. Physical-Design Module Model

To meet the second prerequisite, Cloud Columba models each microfluidic device as a *module*. A module model is a bounding box with pins on its boundaries for inter-module communication. Based on the container category of the device, a module model provides a pre-defined set of potential architectural variants. For example, Figure 3(a) shows the module model for devices that have a ring container. Valves inside the module can be accessed via pins from different boundaries, as shown in Figure 3(b) – (d), supported by the various pre-defined options for control channel routing. Figure 3(e) – (g) show some exemplary physical implementations supported by this module model.

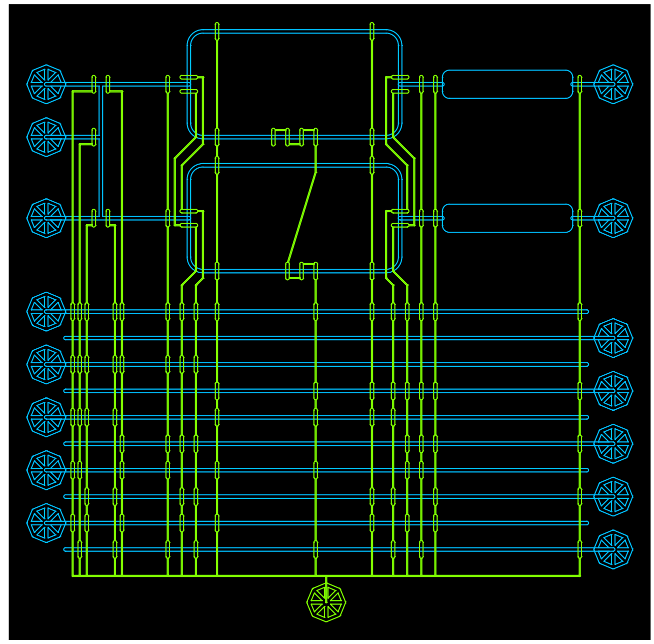

Figure 4: A manufacturing-ready mLSI design synthesized with Cloud Columba. This design consists of one control multiplexer and four devices, i.e., two pressure-sharing (for parallel execution) rings embedded with peristaltic pumps and sieve valves and two pressure-sharing chambers.

Module models allow Cloud Columba to model the multi-layer intra-device synthesis as a discrete pin-selection problem. Rather than explicitly synthesizing the valves and channels inside the modules, Cloud Columba treats a module as a black box and focuses on the placement and routing outside the modules. This abstraction reduces the algorithmic complexity and makes it easier for Cloud Columba to keep up with the continuous technological innovation — whenever new functional components or channel structures are required, the algorithms for physical synthesis can stay the same but only the module models need to be revised.

III. Cloud Columba: The Synthesis Flow

In this section, we demonstrate the usage of Cloud Columba by going through the synthesis flow of a manufacturing-ready mLSI design shown in Figure 4.

Cloud Columba is available at *https://tueieda-columba.srv.mwn.de/*. Figure 5(a) shows an overview of the user interface. It takes three steps for a user to specify all the necessary inputs for the automatic synthesis.

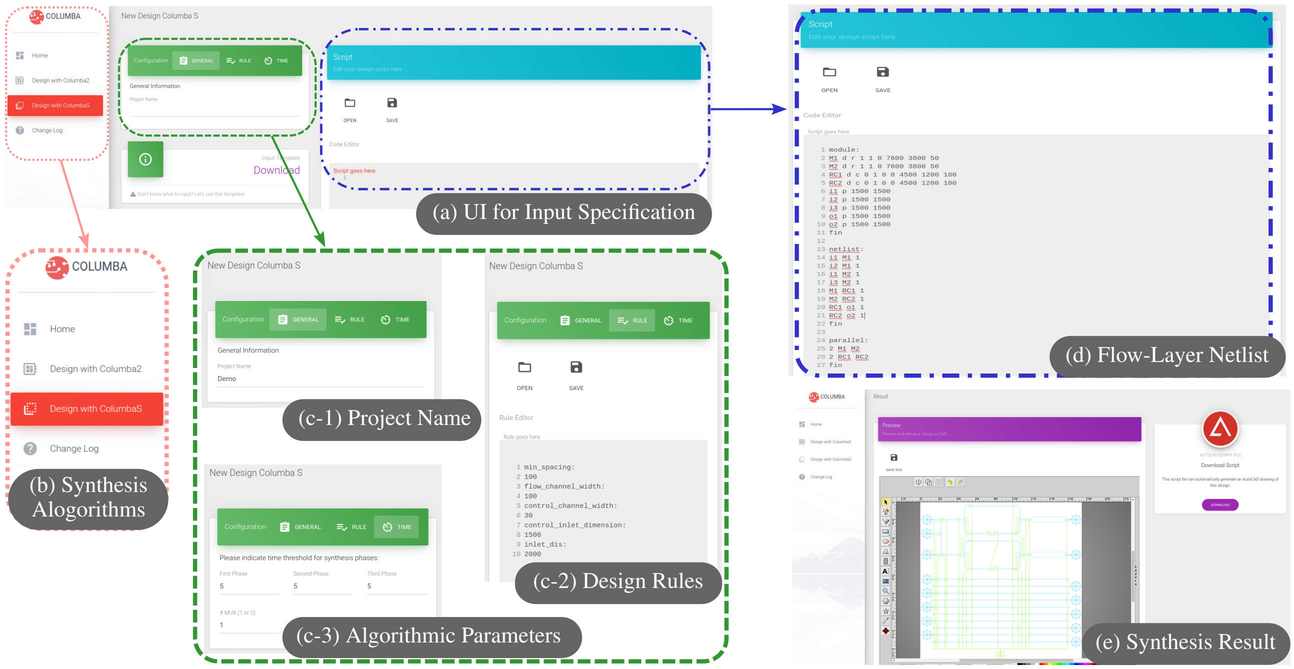

Figure 5: A demonstration of the Cloud Columba synthesis flow. (a) Overview of the user interface for input specification. (b) Navigation bar for selecting the synthesis algorithm. (c) Interface for defining the design parameters. (d) Interface for specifying the flow-layer netlist. Users can either type the netlist in the browser or upload a plain-text file from their devices. (e) Synthesized design, which can be viewed in the browser and downloaded as an .SVG file or an AutoCAD [25] script.

Step 1: Algorithm Selection

The first step is to select an algorithm for synthesis. Cloud Columba provides access to two state-of-the-art synthesis algorithms: Columba 2.0 [21] and Columba S [22], which target different applications. Specifically, Columba 2.0 targets small-scale applications or applications that require a dedicated inlet for each control channel, while Columba S targets large-scale applications or applications that use control multiplexers [2]. To synthesize the design shown in Figure 4, we choose Columba S, as shown in Figure 5(b)

Step 2: Parameter Definition

The second step is to define the project name, the design rules and the algorithmic parameters.

— *Project name*

The project name is a sequence of characters that will be used to name the synthesized design. For example, we name the design 'Demo', as shown in Figure 5(c-1).

— *Design rules*

The design rules specify the geometric constraints (in μm) of the requested mLSI design. Specifically, they specify

1) The minimum spacing between microchannels;
2) The width of flow channels;
3) The width of control channels;
4) The diameter of control inlets; and
5) The minimum center-to-center spacing between inlets.

The design rules from the Stanford Foundry [26] are given by default, as shown in Figure 5(c-2). If users want to make any modification, they can either edit the design rules directly in the browser, or upload a plain-text file specifying the design rules from their local device. To synthesize the design shown in Figure 4, we apply the default design rule.

— *Algorithmic parameters*

Both Columba 2.0 and Columba S synthesize mLSI designs in sequential phases, and the algorithmic parameters specify the time threshold (in seconds) for each optimization phase. The optimization terminates either when an optimal solution is found or when the time threshold is reached. If no feasible solution is found in the given threshold, the optimization continues for another round.

In general, the required optimization time increases with the scale of the design, and Columba S requires less time than Columba 2.0. Considering small-scale designs consisting of e.g. 10 devices, the recommended time threshold in each phase are 10 secs for Columba S and 100 secs for Columba 2.0. More detailed run time analysis can be found in [21] and [22].

Besides, Columba S asks for an additional parameter, namely the number of control multiplexers (# MUX) in the requested design. Compared with 1-MUX designs, 2-MUX designs provide simultaneous control of two valves at the cost of more inlets and larger chip area.

Since the design shown in Figure 4 only consists of 4 devices, we set the time threshold for each optimization phase as 5 seconds, and specify the number of control multiplexers as 1, as shown in Figure 5(c-3).

Step 3: Flow-Layer Netlist Specification

The third step is to specify the flow-layer netlist. Specifically, users need to declare a set of in-/outlets and devices following the concept introduced in Section II and design their logic connections. Besides, devices that are supposed to be executed in parallel can also be specified so that they will share the same control channels.

A detailed instruction for netlist specification can be downloaded together with some exemplary netlists in Cloud Columba. Users can either type the netlist directly in the browser or upload a netlist from their local devices.

To synthesize the design shown in Figure 4, we declare three inlets $i1$, $i2$, $i3$, two outlets $o1$, $o2$, and four devices M1, M2, RC1, and RC2, among which M1 and M2 have rings as containers and are embedded with peristaltic pumps and sieve valves, and RC1 and RC2 have chambers as containers. After that, we specify that $i1$ should be connected to both M1 and M2, and $i2$ and $i3$ should be connected to M1 and M3, respectively. Besides, we also specify that M1, RC1, and $o1$ should be sequentially connected, and M2, RC2, and $o2$ should be sequentially connected. At last, we specify that M1 and M2 should execute in parallel, and RC1 and RC2 should execute in parallel, as shown in Figure 5(d).

After specifying all the necessary inputs within the 3 steps, users can click a "GENERATE DESIGN" button to synthesize their customized mLSI designs. The synthesis result can be previewed in the browser and download as an AutoCAD script file for further modification or fabrication.

Based on the inputs shown in Figure 5(b) – (d), the design shown in Figure 4 can be synthesized in fewer than 3 seconds. Figure 5(e) demonstrates the synthesis result.

IV. Future Improvement

Cloud Columba is a developing project. We are planing a series of updates to improve its capability and accessibility. In this section, we give a brief overview of the planed updates and their potential influences.

A. Module Model Extension and Module Import

Cloud Columba currently provides two module models for general devices, one of which is based on the ring container and the other is based on the chamber container. In the future, we will extend this library by introducing more container categories such as fluid-multiplexers [27], [28], and adding more modification options to the existing module models.

Besides, to keep up with the device-level innovations, we will enable users to import their customized microfluidic designs developed with e.g. AutoCAD [25], SolidWorks [29], 3DμF [30], etc, as modules, which can be automatically reproduced, rotated, and connected with other on-chip components. With the module import function, we want to support intellectual-property-core (IP-core) design for mLSI, where component developers can easily integrate their designs into the system-level synthesis flow.

B. User-Friendly Web Interface

Cloud Columba currently takes a plain-text script for flow-layer netlist specification, which is a programming-like environment that may be inconvenient to researchers outside the computer science community, e.g. bio-engineers and bio-application developers. To make Cloud Columba more user-friendly, we will develop an interactive interface where users can declare their devices and specify their logic connections in a more intuitive and guided way.

V. Conclusion

The mLSI technology has demonstrated their advances in miniaturization, high throughput, and precise control with numerous applications. However, the design of mLSI chips requires exhaustive engineering effort: designers need to specify the three-dimensional geometric features of hundreds to thousands of microfluidic components such as channels and valves, and organize their sophisticated interactions between multiple layers. This time-consuming and error-prone procedure slows down the production of application-specific mLSI designs, and makes it difficult for system-level developers to implement and test their ideas.

To make customized mLSI designs more accessible to a wider scope of users with different backgrounds, we propose Cloud Columba to automate the mLSI design process in an efficient and deterministic manner. Cloud Columba provides a web interface for users to specify the requested devices and their logic connections with simple text inputs, and synthesizes the physical layout of a manufacturing-ready mLSI design automatically within a few minutes or seconds. All the computational tasks are performed on the cloud server, and no specialized interdisciplinary knowledge or powerful local device is required from the users. We will continuously improve Cloud Columba in the future to make it more compatible and user-friendly.

VI. Acknowledgments

The authors would like to thank every member in the Cloud Columba team. Besides Yushen Zhang, who is also an author of this paper, we have Boning Li, Xuesong Zhang, Qingyu Li, Fan Fan, Xiaolin Ma, and Jigao Luo who have contributed to the development of Cloud Columba. The authors would also like to thank their long-term collaborator Prof. Ismail Emre Araci and his group from Santa Clara University for their valuable feedback on Cloud Columba.

References

[1] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000.

[2] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.

[3] J. Liu, M. Enzelberger, and S. Quake, "A nanoliter rotary device for polymerase chain reaction," *Electrophoresis*, vol. 23, pp. 1531–1536, 2002.

[4] F. K. Balagadde, L. You, C. L. Hansen, F. H. Arnold, and S. R. Quake, "Long-term monitoring of bacteria undergoing programmed population control in a microchemostat," *Science*, vol. 309, no. 5731, pp. 137–140, 2005.

[5] J. S. Marcus, W. F. Anderson, and S. R. Quake, "Microfluidic single-cell mRNA isolation and analysis," *Anal. Chem.*, vol. 78, pp. 3084–3089, 2006.

[6] A. R. Wu, J. B. Hiatt, R. Lu, J. L. Attema, N. A. Lobo, I. L. Weissman, M. F. Clarke, and S. R. Quake, "Automated microfluidic chromatin immunoprecipitation from 2,000 cells," *Lab on a Chip*, vol. 9, pp. 1365–1370, 2009.

[7] A. R. Wu, T. L. Kawahara, N. A. Rapicavoli, J. van Riggelen, E. H. Shroff, L. Xu, D. W. Felsher, H. Y. Chang, and S. R. Quake, "High throughput automated chromatin immunoprecipitation as a platform for drug screening and antibody validation," *Lab on a Chip*, vol. 12, pp. 2190–2198, 2012.

[8] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations," in *Proc. Design, Automation, and Test Europe Conf.*, 2016, pp. 624–629.

[9] ——, "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling," in *Proc. Design Autom. Conf.*, 2017, pp. 51:1–51:6.

[10] W. H. Minhass, J. McDaniel, M. Raagaard, P. Brisk, P. Pop, and J. Madsen, "Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 3, pp. 615–628, 2018.

[11] T.-M. Tseng, B. Li, U. Schlichtmann, and T.-Y. Ho, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Des. Test. Comput.*, vol. 32, no. 6, pp. 69–75, 2015.

[12] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, "Transport or store?: Synthesizing flow-based microfluidic biochips using distributed channel storage," in *Proc. Design Autom. Conf.*, 2017, pp. 49:1–49:6.

[13] S. Bhattacharjee, S. Poddar, S. Roy, J. Huang, and B. B. Bhattacharya, "Dilution and mixing algorithms for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 614–627, 2017.

[14] Y. Moradi, M. Ibrahim, K. Chakrabarty, and U. Schlichtmann, "An efficient fault-tolerant valve-based microfluidic routing fabric for droplet barcoding in single-cell analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2018, Early Access.

[15] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, 2015, pp. 141:1–141:6.

[16] T.-M. Tseng, B. Li, M. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis with dynamic device mapping and fluid routing for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 12, pp. 1981–1994, 2016.

[17] M. Shayan, S. Bhattacharjee, Y. Song, K. Chakrabarty, and R. Karri, "Desieve the attacker: Thwarting IP theft in sieve-valve-based biochips," in *Proc. Design, Automation, and Test Europe Conf.*, 2019, pp. 210–215.

[18] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: Co-layout synthesis for continuous-flow microfluidic biochips," in *Proc. Design Autom. Conf.*, 2016, pp. 147:1–147:6.

[19] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai, "Physical co-design of flow and control layers for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1157–1170, 2017.

[20] B. Crites, K. Kong, and P. Brisk, "Diagonal component expansion for flow-layer placement of flow-based microfluidic biochips," *ACM Trans. on Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 126:1–126:18, 2017.

[21] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1588–1601, 2018.

[22] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann, "Columba S: a scalable co-layout design automation tool for microfluidic large-scale integration," in *Proc. Design Autom. Conf.*, 2018, pp. 163:1–163:6.

[23] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and S. R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature Biotechnology*, vol. 22, no. 4, pp. 435–439, 2004.

[24] J. F. Zhong, Y. Chen, J. S. Marcus, A. Scherer, S. R. Quake, C. R. Taylor, and L. P. Weiner, "A microfluidic processor for gene expression profiling of single human embryonic stem cells," *Lab on a Chip*, vol. 8, no. 1, pp. 68–74, 2008.

[25] AUTODESK, *AutoCAD*. http://www.autodesk.com/products/autocad/overview.

[26] Stanford Foundry, *Basic Design Rules*. http://web.stanford.edu/group/foundry.

[27] G. Grossmann, W.-J. Guo, D. W. Ehrhardt, W. B. Frommer, R. V. Sit, S. R. Quake, and M. Meier, "The rootchip: an integrated microfluidic chip for plant science," *The plant cell*, vol. 23, no. 12, pp. 4234–4240, 2011.

[28] M. Meier, R. Sit, W. Pan, and S. R. Quake, "High-performance binary protein interaction screening in a microfluidic format," *Analytical chemistry*, vol. 84, no. 21, pp. 9572–9578, 2012.

[29] SolidWorks, *SolidWorks*. http://www.solidworks.com.

[30] R. Sanka, J. Lippai, D. Samarasekera, S. Nemsick, and D. Densmore, "3DμF-Interactive design environment for continuous flow microfluidic devices," *Scientific Reports*, vol. 9, no. 1, p. 9166, 2019.