

UC Riverside

UC Riverside Previously Published Works

Title

Specification, Integration, and Benchmarking of Continuous Flow Microfluidic Devices:
Invited Paper

Permalink

<https://escholarship.org/uc/item/4cc8t9xb>

ISBN

9781728123509

Authors

Sanka, Radhakrishna

Crites, Brian

McDaniel, Jeffrey

et al.

Publication Date

2019

DOI

10.1109/iccad45719.2019.8942171

Specification, Integration, and Benchmarking of Continuous Flow Microfluidic Devices

Invited Paper

Radhakrishna Sanka
Boston University
Boston, Massachusetts
sanka@bu.edu

Brian Crites
University of California, Riverside
Riverside, California
bcrit001@ucr.edu

Jeffrey McDaniel
University of California, Riverside
Riverside, California
jmcd001@ucr.edu

Philip Brisk
University of California, Riverside
Riverside, California
philip@cs.ucr.edu

Douglas Densmore
Boston University
Boston, Massachusetts
doug@bu.edu

ABSTRACT

The lack of standardization in the specification and representation of microfluidic designs and their corresponding architectures is one of the largest hurdles faced by the developers of Microfluidic Design Automation (MDA) tools. In this paper, we introduce *MINT*, a Microfluidic Hardware Description Language (MHDL) for defining components and devices in a human readable manner, and *ParchMint*, an MDA interchange format and associated benchmark suite that can be used to compare the performance of different physical design algorithms. We further demonstrate how the introduction of *MINT* and *ParchMint* into the engineering workflow can bridge the gaps from the specification to the fabrication of microfluidic devices. While recent efforts to democratize microfluidics have been recognized by the community, there is an unfortunate lack of open source tools, design languages, and standards. Consequently, microfluidic designs shared on open platforms such as Metafluidics[15] leave conceptual gaps in terms of missing design information that are necessary to realize the “creative process flows” (Reproduce, Remix, and Test multiple systems). *MINT* and *ParchMint* are open source projects, which allows the community to contribute and extend their functionality to enable advanced algorithmic methodologies and new commercialization possibilities that differ from the vertically integrated industries we see today.

KEYWORDS

Microfluidics, Design Automation, Standards, Benchmarking

1 INTRODUCTION

The potential to integrate Laboratory-on-a-chip (LoC) devices to increase the throughput and repeatability of laboratory experiments has spurred research efforts toward Microfluidic Design Automation (MDA) [2]; while algorithmic progress has been reported, there has been an abject failure at the community level to integrate these algorithms into practical tools that are usable by microfluidic designers. One major impediment is the lack of standardized design specification systems and interface formats that describe a sufficient set of physical parameters to facilitate direct and unbiased comparison among academic MDA software artifacts, coupled with

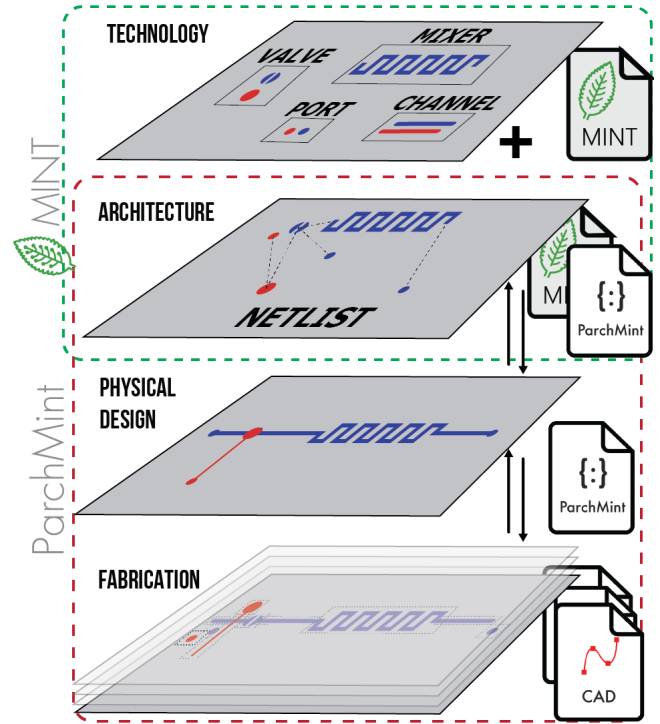


Figure 1: The four levels of abstraction in the continuous flow microfluidic physical design generation process that is captured by *MINT* and *ParchMint*. *MINT* captures the design of a microfluidic device in the TECHNOLOGY and ARCHITECTURE levels in a human-readable format that encapsulates the designer’s perspective. The *ParchMint* standard interchange format allows for the design captured in *MINT* to be converted into a machine-readable format appropriate for PHYSICAL DESIGN algorithms, and artwork generation algorithms at the FABRICATION level.

a path to fabrication to allow for experimental validation of these designs in a laboratory setting.

To address these limitations, this paper introduces *MINT*¹, a Hardware Description Language (HDL) that describes a microfluidic device as a netlist comprising a collection of tunable parametric components in a human readable format, and *ParchMint*², an MDA interchange format and associated comprehensive microfluidic benchmark suite that is intended for use by the design automation community. *MINT* and *ParchMint* are open standards that aim to reduce the barriers for integrating research in MDA, fostering collaborations, innovation and to help facilitate non-experts to create designs for different applications.

2 FRAGMENTED RESEARCH WORKFLOWS

Ongoing research efforts in MDA employ a small collection of microfluidic architectural netlists, summarized in Table 1, which are used as a basis for algorithmic comparison. These benchmarks have been generated either by converting bioassay specifications into a microfluidic netlist capable of executing the specified assay or by hand-converting published microfluidic chips into netlists based on manual inspection of published device images. To date, there has been little to no consistency among research teams in terms of these efforts, making it difficult to perform direct comparisons.

The most popular benchmark sets that are or were publicly available are two suites of bioassay specifications for electrowetting-based digital microfluidics, one released by DTU Compute in Denmark [1] and another by Duke University in the United States³. Over the years, these have become the de-facto “standard” benchmark suites for continuous-flow based microfluidics and constitute approximately half of benchmarks in literature today. Several papers have introduced algorithms to convert these bioassay specifications into continuous flow netlists [14, 18, 25, 32], which can then be used for studies in physical design. However, the netlists generated by these tools have not been shared, and are likely inconsistent in terms of underlying assumptions, for example, regarding component geometries. Further, to the best of our knowledge, no continuous flow device based on any of these netlists has been validated in a wet laboratory setting by an experimentalist.

Even when there is a desire to use the same benchmarks across groups, there is no publicly available set of architectural netlists or agreed upon file format for specifying them. For example, McDaniel et al. uses *Synthetic 1-5 (10-50)* when reporting results for a simulated annealing based placement method [23], Wang et al. [37] builds on this work by introducing a negotiation based router and a placement adjustment system and compares against McDaniel et al.[23]. However, when making the comparison Wang et al.[37] used the Duke University benchmark to generate the netlist, rather than the DTU Compute benchmark set used by McDaniel et al.[23], making a direct comparison of the two methods impossible. Compounding this issue is the lack of specific information given for each benchmark. McDaniel et al.[23] gives no additional information on the benchmarks used, and Wang et al.[37] only provides the number of components that each benchmark contains. It is impossible to discern whether the improvements reported in the latter paper

Table 1: Comparison of the benchmarks used for the evaluation of different physical design algorithms in literature

Benchmark	Minhass et al. [25]	McDaniel et al. [23]	Yao et al. [40]	Wang et al. [37, 38]	Tseng et al. [33, 34]	Grimmer et al. [9]	Crites et al. [5, 6]
EA [25]*	✓						
PCR [24, 31]*	✓	✓		✓		✓	
IVD [24]*	✓						
CPA [24]*	✓						
Synthetic 1(10) [24]*	✓	✓	✓				✓
Synthetic 2(20) [24]*		✓	✓				✓
Synthetic 3(30) [24]*	✓	✓	✓				✓
Synthetic 4(40) [24]*	✓	✓	✓				✓
Synthetic 5(50) [24]*	✓	✓	✓				✓
ProteinSplit-1 [31]*				✓		✓	
ProteinSplit-2 [31]*				✓		✓	
InVitro-1 [31]*				✓		✓	
InVitro-2 [31]*				✓		✓	
InVitro-3 [31]*				✓		✓	
Kinase act. [8]					✓		
Acid proc. [11]					✓		
mRNA iso [19]					✓		
Gradient Generator [29]							✓
HIV1 [17]							✓
ChIP (4IP) [39]					✓		
ChIP (10IP)					✓		
Chip1			✓				
Chip2			✓				
Cell free Bio Net [26]					✓		
AquaFlex-3b †							✓
AquaFlex-5a †							✓
*interpreted from a bioassay application							
†Proprietary designs from Microfluidic Innovations LLC							
PCR is listed in the DTU and Duke benchmark suites							

are due to improvements in physical design algorithms, differences in the netlists, or a combination of both factors.

The literature on microfluidics typically presents devices that were designed manually, e.g., using software such as AutoCAD, fabricated, and then used to perform a variety of biological assays. Some of these publications provide chip photos or schematic layouts with nominally sufficient detail to allow MDA researchers to derive netlists through visual inspection; however, this obscures the engineering that goes into the design and validation of ‘realistic’, ‘working’ microfluidic devices. Moreover, there is a high amount of variability in interpretation: many MDA systems represent components using a bounding box, there can be substantial variations in the measured size of the component as obtained from the source material. Additionally, the decomposition of a “flattened” network of valves and channels into a netlist of components also varies, as

¹ <https://github.com/CIDARLAB/MINT/wiki>

² parchmint.org

³ Benchmarks for digital microfluidic biochip design and synthesis - F. Su and K. Chakrabarty. 2006. (URL no longer active)

there exists no deterministic algorithm to select which valves and routes should be combined to create one component. Finally, there are also differences in the way group represent their components. For example, Grimmer et al.[9] assumes each component has four terminals located at each corner, while others assume that terminals can be located at other locations on a component's perimeter, as illustrated in Figure 3. The granularity of the device specification can also affect algorithmic performance and the efficacy of algorithms that aim to generate "working designs".

3 MINT

HDL-based design methodologies have been a critical driver in the world of electronics. By allowing different designs to be captured in a standardized format, HDLs formally capture specifications while allowing designers to track changes and employ automated toolchains to generate physical designs, validate against design rules and rapidly iterate through variations to get functional designs. HDLs also abstract some of the complexity associated with the design specification and help to reduce the effort required to engineer a semiconductor device. Similar design methodologies and abstractions could benefit the MDA research community.

Fluigi [12], an end-to-end design framework that generates microfluidic devices from a netlist specification, first introduced *MINT* as a netlist standard. Motivated by MHD [20], which was a purely structural specification, *MINT* separated the structural and behavioral aspects of the design. This allowed researchers to specify generic designs that are not tied to a specific assay or netlist. Additionally *MINT* captured the geometric properties of microfluidic components (Figure 3) that constitute the design. This allowed microfluidic designers to perturb the physical properties of components to optimize and tune the performance of microfluidic designs without modifying the netlist.

3.1 Parametric Component Library

The heart of *MINT* is a parametric component library that abstracts the functional units of a microfluidic netlist into components that can be used across microfluidic applications. Each component described by *MINT* can have an arbitrary number of input and output ports at any position on the component perimeter. This enables abstraction of components and netlists that incorporate both flow and control elements and the specification of components defined by adjustable geometric parameters that define its size, performance, and functional characteristics.

3.2 Logical Layers

Today, most MDA tools target continuous flow microfluidic devices fabricated using soft-lithography, which creates molds that are used to imprint channel structures onto a flexible PDMS substrate via molds [28]. Multi-layer soft lithography [35] enables the creation of valves at intersection points between channels on adjacent layers, along with other control-related infrastructure; separate PDMS molds are used to create the *FLOW* and *CONTROL* layers. Monolithic membrane valves [10] are an alternative design that sandwiches a PDMS membrane between two rigid substrates (e.g., glass) into which flow and control channels have been etched using standard microfabrication techniques; the key advantage here

is the elimination of the need to create expensive molds to imprint patterns in PDMS. More recently, low-cost rapid prototyping technologies have emerged [16, 36] which can carve channels into rigid substrates without the need for more expensive microfabrication equipment; monolithic membrane valves can easily be constructed using rapid prototyping technologies as well. However advent of newer manufacturing processes requires any HDL describing microfluidics to be agnostic to them and allow for a standardized way of describing microfluidic devices.

The user constructs a device using *MINT* by specifying separate *FLOW* and *CONTROL* logical layers without explicitly specifying the order and relationships between them. For example, the device shown in Figure 2 A. has one *FLOW* and one *CONTROL* layer; this example shows how *MINT*'s syntax permits the user to declare a new logical layer by using the "LAYER (FLOW|CONTROL)" and "END LAYER" blocks. The "Logical Layers" abstraction allows *MINT* to be agnostic to technology choices and their manufacturing requirements. The underlying software can easily generate the design output for each 'Manufacturing Layer' (a 2D or 3D design file) which is used to manufacture the microfluidic chip [30].

3.3 Device Descriptions using MINT

Microfluidic designers specify the structure of their chips in terms of components and connections using *MINT*. For example, Figure 2 A. depicts a simple design that incorporates a *MIXER* component with an input valve that regulates the flow of the fluid that enters the mixer. The design specification begins with a 'DEVICE' statement, followed by the descriptions of each *logical layer*, 'FLOW' and 'CONTROL' in this case. In Figure 2 A., the specification includes components in lines 5 – 6 and 16 – 17 along with interconnections (using the 'CHANNEL' statement) in lines 9 – 10 and 18. *MINT* also provides syntactic constructs for constraints, relationships between components (e.g., valve *v1* is placed on channel *c1* using the 'on' keyword) and parameters (camel-cased with values following the component/connection definition) for each of the components in the design description. Detailed tutorials and additional examples are available on the *MINT* wiki¹.

3.4 Extensibility

MINT enables specification of microfluidic device architectures with layout constraints in a manner that can easily be interpreted by physical design software. To compile a *MINT* file, the designer initializes a database of *Technology Files* which describes *MINT*'s component library and provides the *MINT* compiler with sufficient information to generate the physical design of the device. The *MINT* format is designed to be readable by any physical design tool and is not tied to any specific algorithmic choices or strategies.

More information on constraints, syntax, component *Technology File* formats are available on the *MINT* wiki¹. To facilitate extensibility and portability, the *MINT* compiler's grammar was specified using ANTLR[27], which automatically generates the lexer and parser; this reduces the effort required to extend or contribute to the standard. A Java implementation of the *MINT* compiler is available on Github¹.

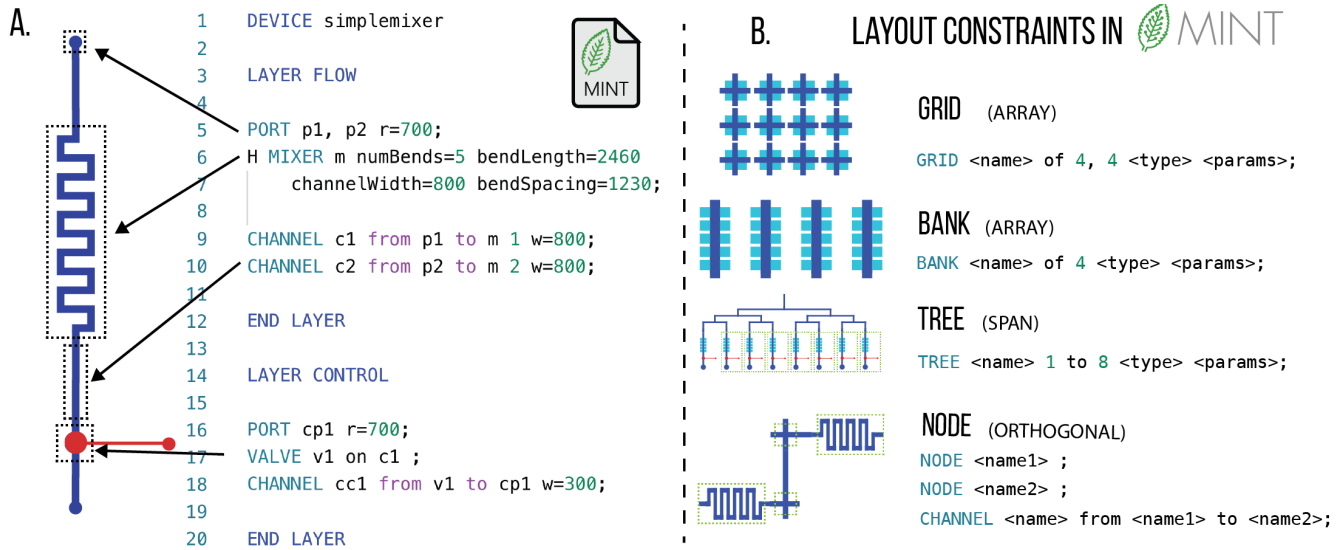


Figure 2: MINT is a hardware description language used to define microfluidic architectures and their corresponding layout constraints using the built-in grammar. **A.** shows an example of ‘simplemixer’, where a MIXER connected to two PORTS using a VALVE is defined using MINT. MINT not only describes the connectivity but also the geometric properties of each of the components that would dictate the complexity of the component and whether the components and connections belong in the FLOW/CONTROL layers. **B.** shows examples how different syntaxes used to describe the structure of the microfluidic automatically generate layout constraints that can be utilized by the physical design generation algorithms to capture better the designer’s intent and expertise in creating the final design layouts.

3.5 Conversion to ParchMint

MINT was primarily designed to be a human readable component and device description language. This means the format was not designed to contain intermediate placement and routing information utilized by MDA systems. Without this ability the MINT standard itself cannot facilitate work across separate teams and tools nor can it allow for direct comparison between different designs. Because of this, the MINT compiler converts the MINT description to a ParchMint file for use by physical design software that generates the final layout, encoding the necessary information from MINT into the proper ParchMint fields as described in Section 4.1.

4 PARCHMINT

4.1 Standard Interchange Format

ParchMint is a standard interchange format that describes the architectural netlist of continuous flow microfluidic devices. Standardizing the interchange format will allow research groups that independently develop their own devices to share their netlists, increasing collaboration and enabling unbiased comparison of physical design algorithms. ParchMint netlists are specified as JavaScript Object Notation (JSON) files, which itself is a standard format for structured data, which can easily be parsed in any programming language. The ParchMint benchmarks, which are described later in Section 4.2, include a publicly described JSON-Schema file² that specifies the requisite fields; new netlists contributed by others can be verified against the schema for correctness.

The ParchMint architecture netlist is specified by three top-level object lists: *layers*, *components*, and *connections*. Each object within the system has a name and a unique ID field: the name field is human readable, while the ID field is referenced by other objects. The layers top-level object list can contain any number of layer JSON objects. The majority of fabricated microfluidic devices today have at most two layers, one for fluid flow and the other for control.

Each component JSON object contains an entity field that encodes the specific MINT component that it represents, a layer list that specified which layers comprise it, *x*- and *y*-span field that specify a 2D bounding box surrounding the component’s internal geometry, and a list of ports. Each port includes an (*x*, *y*) location value that represents its location on the perimeter of the component bounding box, along with a label which uniquely identifies it within that particular component.

Connection JSON objects, which represent fluid channels, contain a layer list along with a single source and a list of one or more sinks. Each source and sink contains a field that references the ID of a component connected to a channel, and, if needed, the ID of the connecting port. Multi-sink connection objects can specify channels that provide direct fluidic connections between multiple components.

The ParchMint standard includes optional geometric information that can specify the physical layout of a microfluidic netlist after placement and routing; this serves to facilitate external evaluation and validation. Research groups that study physical design algorithms can publish placed-and-routed netlists in the ParchMint

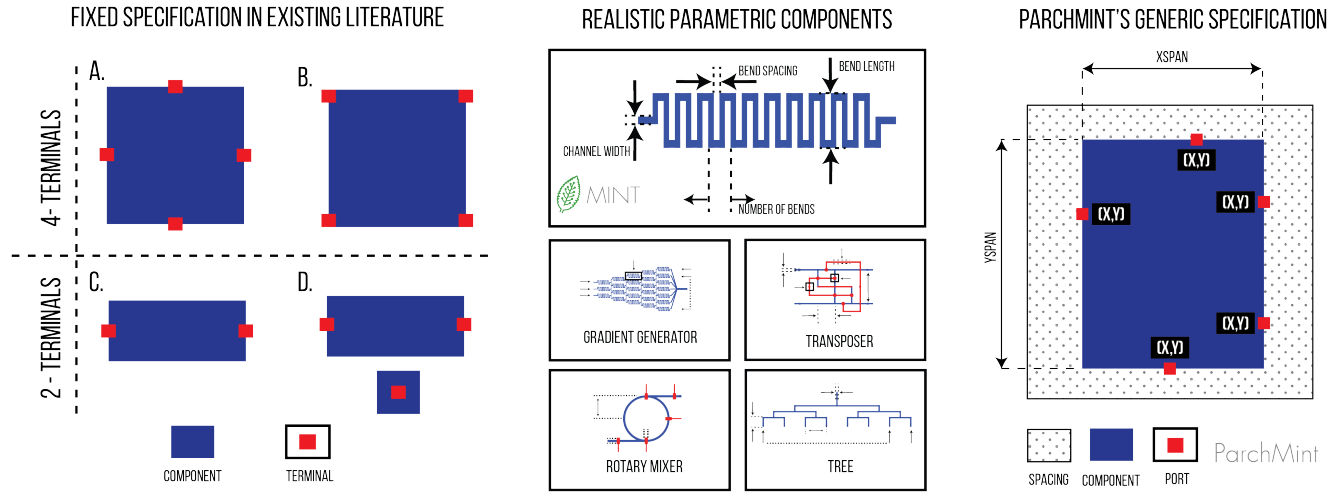


Figure 3: The figure shows how the component representations in Microfluidic Design Automation (MDA) literature is inadequate for the practical use and how the component representation in *ParchMint* can bridge that gap for practical usage. The panel on the left shows how MDA systems represent the components that are used for placement and routing in varying ways. Different systems assume fixed positions for terminals where the channels can connect (red) on the components' boundary (blue). (a-b) Four terminals may be assumed either at the corners of the bounding box or at the center of each bounding box wall [9, 22, 37] (c-d) Two terminals are assumed for each component existing at the center of two opposing bounding box edges, with the latter including a single terminal at the component's center [7, 34]. These variations in the basic component models induce variability in the layouts generated by the algorithms. The panel in the center shows how the realistic components used in microfluidic design cannot be approximated to the common abstractions used in literature. The panel on the right shows how the component representation in *ParchMint* can allow for variation in the number/position of terminals, size and allows for design rules such as component spacing.

format to enable direct and unbiased comparisons between different physical design algorithms. The *features* object list at the top level specifies the concrete placement and routing information of components and connections to support design-rule checking and evaluation of algorithmic post-processing steps. This list contains two types of objects, *component features* and *connection features*. Each component can have at most one component feature which specifies the location where it is placed; connection features contain begin and endpoints for each straight-line segment of each connections, and allows multiple straight-line segments to constitute a single connection object in the connection list.

Using JSON allows researchers to extend the *ParchMint* format with additional information in the form of *custom* and *params* fields as seen in Figure 4. For example, these fields could be used to encapsulate information such as emerging layout constraints, custom design rules [12], specific component/connection physical design information, and protocol/assay descriptions [38, 40] that may be needed to support specific biological applications. Further, the *custom* and *params* fields can encode any additional values needed to ensure correct post-processing or netlist validation of a netlist. Any design algorithms that require external values to generate a specific result netlist should encode those values within the *params* field so that future designers can easily reuse and validate results.

4.2 Comprehensive Benchmark Suite

The *ParchMint* benchmarks² are a set of publicly released planar undirected multigraphs that are agnostic to existing physical design algorithms while being representative of real-world microfluidic designs used to conduct biological experiments. Most benchmarks currently being used to evaluate academically-developed physical design algorithms consist of no more than 100 components. A primary claim of Microfluidic Very Large Scale Integration (mVLSI) is that one can fit thousands of valves onto a single device [3]. The *ParchMint* benchmarks are based on real-world devices that have been scaled up to more than 400 components (refer Table 2; with community assistance, we hope to introduce even larger representative benchmarks in the future.

Assay Inspired Benchmarks. These netlists were created by examining images of microfluidic devices that were published in peer-reviewed literature. While these devices can execute an end-to-end biological assay, it was not always possible to derive the actual assay specification from the publication. We manually translated each image to a planar netlist in the *ParchMint* format. We manually inferred components by identifying repeated patterns in the physical layout, and applied a general set of component descriptions across the entire architecture.

USING *ParchMint* BENCHMARK SPACES AS A TOOL TO FOR DEVELOPING NEW LAYOUT ALGORITHMS

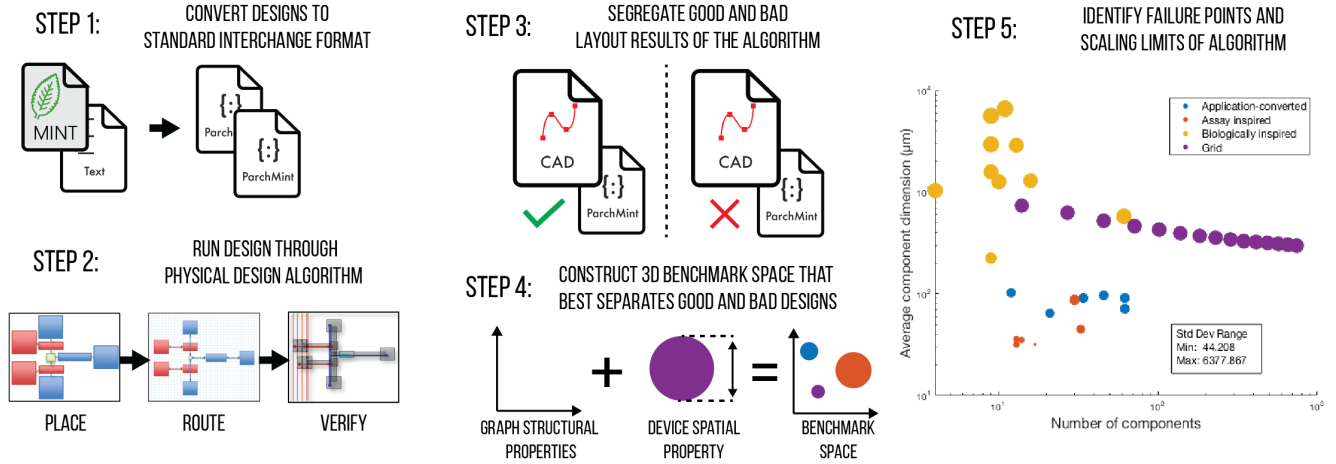


Figure 4: *ParchMint* can be used to examine the limits of physical design algorithms. The above figure shows one such workflow where algorithm designers can construct benchmark spaces using the data captured by designs following the *ParchMint* specification and understand the performances of physical design algorithms on benchmarks located in different regions of the benchmark spaces. This allows researchers to identify the shortcomings of algorithms and approaches applied to generate the physical design of the microfluidic device.

Application-converted Benchmarks. These netlists, **synthetic_N** (1-7), were derived from a set of bioassay specifications originally released by DTU Compute [24]. These assays were originally designed for electrowetting-based digital microfluidics, and were converted to netlists using an architectural synthesis heuristic [25]. Some of the netlists that were generated using this method were non-planar; we manually planarized these netlists by manually removing valves and channels until the netlist became planar.

Biologically Inspired Benchmarks. These netlists manually designed using the published device images used in synthetic biology experiments surveyed by Huang and Densmore [12, 13]. The designs were originally defined using *MINT*, which subsequently converted *ParchMint* files.

Grid Benchmarks. These netlists include designs **grid_N** (2-16) which contain generic $N \times N$ grids of cell-traps with valves that allow for the selection of either horizontal or vertical rows.

4.3 Benchmarking Case Study

We synthesized the flow layers of the *ParchMint* benchmarks using physical design systems developed independently by two research groups. The discussion that follows highlights the issues that were uncovered during this process, which illustrate the need for a comprehensive microfluidic benchmark suite that includes scale and feature variance; otherwise, it will not be possible to properly evaluate and compare different physical design automation algorithms.

Both systems were able to automatically generate valid placements for all benchmarks; however, valid routing results could not always be obtained. For example, a simulated annealing placer that used Hadlock’s Algorithms for channel routing [12] could not find

routable layouts for the *Assay Inspired* and *Application-converted* benchmarks.

The Planar Embedding and Network-Flow algorithm[21] successfully placed and routed all in the *Biologically Inspired Benchmarks* except for *net_mux*, which contains two components with very high connectivity: 19 connections in each case. The presence of a large number of connections to a single component created areas of high routing density, which became difficult to resolve. Placement via planar graph embedding exacerbates the situation since it does not consider the possibility to rotate the orientation of components during placement. One of the high-connectivity components has a majority of ports on one of its sides: if that component’s ports face away from the majority of its neighboring components, then it becomes increasingly difficult to obtain a legal routing result.

The Planar Embedding and Network-Flow method was only able to route *grid_2* in the *Grid Benchmarks* suite. The problem is that the placements, which were otherwise legal, were not well-attuned to creation of a grid-like routing structure. This is a byproduct of the Chrobak-Payne planar embedding algorithm [4], which creates an initial placement. This method employs triangulation to place initial points, which are later expanded to 2D components, within a 2D plane. The general tendency is to place components within sub-triangles along the diagonal of the plane. This arrangement leads to high congestion and a high rate of routing failure along the diagonal.

The Planar Embedding and Network-Flow algorithm successfully placed and routed all of the *Assay Inspired* and *Application Converted* benchmarks; however, simulated annealing placement coupled with Hadlock’s routing algorithm were less successful. The simulated annealing placer minimizes a cost function based on total connection length and total chip area: after extensive testing,

Table 2: Benchmarking failures for Planar Embedding & Network-flow and Simulated Annealing & Hadlocks Algorithms

Layout Algorithms		Planar Embedding and Network-Flow				Simulated Annealing and Hadlocks			
Coverage		6/6	7/7	9/10	1/16	0/6	0/7	10/10	12/16
Failure Modes	Placement Failure					●	●		
	High Density Component			●					
	Poor Embedding				●				
	Memory Limit								●
Collection		Assay Inspired	Synthetic	Biologically Inspired	Grid	Assay Inspired	Synthetic	Biologically Inspired	Grid
Design Complexity		# Components [min, max]				[13, 33]	[12, 62]	[4, 61]	[14, 414]
		# Connections [min, max]				[12, 38]	[11, 64]	[3, 87]	[14, 566]

we observed that this strategy led to compact layouts which were non-planar and difficult to route. These issues were exacerbated by: 1) small component sizes, 2) sparse connectivity, and 3) the lack of consideration for routability in the objective function.

Another observable difference arises from the amount of buffer space that is reserved around components and connections for routing. For example the *simple* and *flow_focus* benchmarks from the *Biologically Inspired Benchmarks* set exhibit a relatively large amount of buffer space surrounding relatively small component sizes, which wastes space when Hadlock’s algorithm is used to route channels. In contrast, the Network-Flow router reserves far less buffer space, but aggressively rips up, re-orders, and re-routes for multiple iteration before it finds a valid routing solution. The amount of buffer space that was allocated was independently hard-coded in the independently-developed physical design tools; an ideal comparison would be performed either using the same amount of buffer space, or repeatedly varying the buffer space for each algorithm until a valid routing solution can be found. This issue highlights difficulties that may occur when research teams exchange code and benchmarks, even when using a standard interchange format between physical design step. This type of variability between methods necessitates that inclusion of the *params* field in the standard interchange format.

Both flow-layer routers evaluated here are grid-based. Fine-granularity routing can therefore stress the memory capacity of a modern computer for benchmarks of non-trivial size. Benchmarks exhibiting sparse placements that entail a substantial allocation of whitespace required more memory than was available to allocate a full routing grid. To compensate, both Hadlock’s router and the Network-Flow router employ a variable precision routing grid (e.g., 1 μ m, 10 μ m, or 100 μ m) using a pre-processing method that scales down the entire netlist geometry by a factor prior to placement step. Although scaling decreases the accuracy of the results by a small amount, it allows us to place and route much larger benchmarks. Many of the benchmarks presented here were processed with a 10 μ m routing grid, and are annotated in Table 2.

The two routers that were compared used different methods to determine the start and ending point of each channel. While some components have ports that are functionally interchangeable, others require specific fluid channels to be routed to specific ports. The Network-flow Algorithm assumes the ability to freely assign ports to routed channels, while Hadlock’s router requires all ports to be pre-assigned prior to routing. To eliminate this discrepancy, we

pre-assigned ports to all connections in the benchmark netlist specification, even in cases where components do not require specific port assignments.

5 CONCLUSION

We hope that the *MINT* and *ParchMint* file formats and benchmarks spur subsequent standardization efforts to the benefit of the MDA research community. As a specification language, *MINT* effectively captures microfluidic device architectures in a format that is both human and machine readable, while *ParchMint* encompasses information relevant to physical design. Members of the MDA community can contribute to both *MINT* and *ParchMint* standards via their respective Github Repositories¹². Our evaluation of two independently-developed physical design software packages was non-exhaustive, but uncovered many non-obvious challenges that must be overcome to enable direct and unbiased comparisons between design algorithms in the future and demonstrates the need for the MDA community to communicate and collaborate effectively.

We hope that future microfluidic designers would be able to specify their designs using *MINT* to capture the design parameters that are necessary to ensure functionality, that researchers who develop new MDA algorithms will evaluate them using *ParchMint* and that widespread adoption of these standards will reduce the barrier to reporting algorithmic developments and being able to generate functional device designs.

ACKNOWLEDGMENTS

We would like to thank Joshua Lippai for his contributions to *ParchMint* Standard and for his help in creating the benchmark space visualizations. This work is supported in part by NSF Award Numbers #1351115, #1536026, #1522074 and #1640757.

REFERENCES

- [1] 2012. Microfluidic Flow-Based Biochips Benchmark Suite. <https://sites.google.com/site/mlsibiochips/home>
- [2] Ismail Emre Araci and Philip Brisk. 2014. Recent developments in microfluidic large scale integration. *Current Opinion in Biotechnology* 25 (Feb. 2014), 60–68. <https://doi.org/10.1016/j.copbio.2013.08.014>
- [3] Ismail Emre Araci and Stephen R Quake. 2012. Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves. *Lab on a Chip* 12, 16 (2012), 2803–2806.
- [4] Marek Chrobak and Thomas H Payne. 1995. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.* 54, 4 (1995), 241–246.

- [5] Brian Crites, Karen Kong, and Philip Brisk. 2017. Diagonal Component Expansion for Flow-Layer Placement of Flow-Based Microfluidic Biochips. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 126.
- [6] Brian Crites, Karen Kong, and Philip Brisk. 2017. Reducing Microfluidic Very Large Scale Integration (mVLSI) Chip Area by Seam Carving. In *Proceedings of the on Great Lakes Symposium on VLSI 2017 (GLSVLSI '17)*. ACM, New York, NY, USA, 459–462. <https://doi.org/10.1145/3060403.3060461>
- [7] M. C. Eskesen, P. Pop, and S. Potluri. 2016. Architecture synthesis for cost-constrained fault-tolerant flow-based biochips. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 618–623.
- [8] Cong Fang, Yanju Wang, Nam T. Vu, Wei-Yu Lin, Yao-Te Hsieh, Liudmilla Rubbi, Michael E. Phelps, Markus Mäijschen, Yong-Mi Kim, Arion F. Chatzioannou, Hsian-Rong Tseng, and Thomas G. Graeber. 2010. Integrated Microfluidic and Imaging Platform for a Kinase Activity Radioassay to Analyze Minute Patient Cancer Samples. *Cancer Research* 70, 21 (Nov. 2010), 8299–8308. <https://doi.org/10.1158/0008-5472.CAN-10-0851>
- [9] Andreas Grimmer, Qin Wang, Hailong Yao, Tsung-Yi Ho, and Robert Wille. 2017. Close-to-optimal placement and routing for continuous-flow microfluidic biochips. In *22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017, Chiba, Japan, January 16-19, 2017*. IEEE, 530–535. <https://doi.org/10.1109/ASP-DAC.2017.7858377>
- [10] William H Grover, Alison M Skelley, Chung N Liu, Eric T Lagally, and Richard A Mathies. 2003. Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices. *Sensors and Actuators B: Chemical* 89, 3 (April 2003), 315–323. [https://doi.org/10.1016/S0925-4005\(02\)00468-9](https://doi.org/10.1016/S0925-4005(02)00468-9)
- [11] Jong Wook Hong, Vincent Studer, Giao Hang, W. French Anderson, and Stephen R. Quake. 2004. A nanoliter-scale nucleic acid processor with parallel architecture. *Nature Biotechnology* 22, 4 (April 2004), 435–439. <https://doi.org/10.1038/nbt951>
- [12] Haiyao Huang. 2016. *Fluidig: An End-to-End Software Workflow for Microfluidic Design*. Ph.D. Boston University. <https://doi.org/10.1017/CBO9781107415324.004> arXiv:arXiv:1011.1669v3
- [13] Haiyao Huang and Douglas Densmore. 2014. Integration of microfluidics into the synthetic biology design flow. 14, 18 (Aug. 2014), 3459–3474. <https://doi.org/10.1039/C4LC00059K>
- [14] Wei-Lun Huang, Ankur Gupta, Sudip Roy, Tsung-Yi Ho, and Paul Pop. 2017. Fast architecture-level synthesis of fault-tolerant flow-based microfluidic biochips. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, David Atienza and Giorgio Di Natale (Eds.). IEEE, 1667–1672. <https://doi.org/10.23919/DATE.2017.7927262>
- [15] David S. Kong, Todd A. Thorsen, Jonathan Babb, Scott T. Wick, Jeremy J. Gam, Ron Weiss, and Peter A. Carr. 2017. Open-source, community-driven microfluidics with Metafluidics. *Nature Biotechnology* 35, 6 (June 2017), 523–529. <https://doi.org/10.1038/nbt.3873>
- [16] Ali Lashkaripour, Ryan Silva, and Douglas Densmore. 2018. Desktop micromilled microfluidics. *Microfluidics and Nanofluidics* 22, 3 (26 Feb 2018), 31. <https://doi.org/10.1007/s10404-018-2048-2>
- [17] Baichen Li, Lin Li, Allan Guan, Quan Dong, Kangcheng Ruan, Ronggui Hu, and Zhenyu Li. 2014. A smartphone controlled handheld microfluidic liquid handling system. *Lab Chip* 14 (2014), 4085–4092. Issue 20. <https://doi.org/10.1039/C4LC00227J>
- [18] Mengchu Li, Tsun-Ming Tseng, Bing Li, Tsung-Yi Ho, and Ulf Schlichtmann. 2016. Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations. In *2016 Design, Automation & Test in Europe Conference & Exhibition, DATE 2016, Dresden, Germany, March 14-18, 2016*, Luca Fanucci and Jürgen Teich (Eds.). IEEE, 624–629. <http://ieeexplore.ieee.org/document/7459386/>
- [19] Joshua S. Marcus, W. French Anderson, and Stephen R. Quake. 2006. Microfluidic Single-Cell mRNA Isolation and Analysis. *Analytical Chemistry* 78, 9 (May 2006), 3084–3089. <https://doi.org/10.1021/ac0519460>
- [20] J. McDaniel, A. Baez, B. Crites, A. Tammewar, and P. Brisk. 2013. Design and verification tools for continuous fluid flow-based microfluidic devices. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*. 219–224. <https://doi.org/10.1109/ASP-DAC.2013.6509599>
- [21] J. McDaniel, B. Crites, P. Brisk, and W. H. Grover. 2015. Flow-Layer Physical Design for Microchips Based on Monolithic Membrane Valves. *IEEE Design Test* 32, 6 (Dec. 2015), 51–59. <https://doi.org/10.1109/MDAT.2015.2459699>
- [22] Jeffrey McDaniel, Brian Crites, Christopher Curtis, and Philip Brisk. 2015. Design Automation for Flow-based Microfluidic Biochips. In *37th Annual International conference of the IEEE Engineering in Medicine and Biology Society*.
- [23] Jeffrey McDaniel, Brendon Parker, and Philip Brisk. 2014. Simulated annealing-based placement for microfluidic large scale integration (mLSI) chips. *2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)* 4 (oct 2014), 1–6. <https://doi.org/10.1109/VLSI-SoC.2014.7004170>
- [24] Wajid Hassan Minhass, Paul Pop, and Jan Madsen. 2011. System-level modeling and synthesis of flow-based microfluidic biochips. In *Proceedings of the 14th international conference on Compilers, architectures and synthesis for embedded systems*. ACM, 225–234.
- [25] Wajid Hassan Minhass, Paul Pop, Jan Madsen, and Felician Stefan Blaga. 2012. Architectural Synthesis of Flow-based Microfluidic Large-scale Integration Biochips. In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES '12)*. ACM, New York, NY, USA, 181–190. <https://doi.org/10.1145/2380403.2380437>
- [26] Henrike Niederholtmeyer, Viktoria Stepanova, and Sebastian J. Maerkl. 2013. Implementation of cell-free biological networks at steady state. *Proceedings of the National Academy of Sciences* 110, 40 (Oct. 2013), 15985–15990. <https://doi.org/10.1073/pnas.1311166110>
- [27] Terence Parr and Kathleen Fisher. 2011. LL(*): The Foundation of the ANTLR Parser Generator. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11)*. ACM, New York, NY, USA, 425–436. <https://doi.org/10.1145/1993498.1993548>
- [28] Dong Qin, Younan Xia, and George M. Whitesides. 2010. Soft lithography for micro- and nanoscale patterning. *Nature Protocols* 5, 3 (March 2010), 491–502. <https://doi.org/10.1038/nprot.2009.234>
- [29] Minsoung Rhee and Mark A. Burns. 2008. Microfluidic assembly blocks. *Lab Chip* 8 (2008), 1365–1373. Issue 8. <https://doi.org/10.1039/B805137B>
- [30] Radhakrishna Sanka, Joshua Lippai, Dinihi Samarasekera, Sarah Nemsick, and Douglas Densmore. 2019. 3Dliff - Interactive Design Environment for Continuous Flow Microfluidic Devices. *Scientific Reports* 9, 1 (Dec. 2019). <https://doi.org/10.1038/s41598-019-45623-z>
- [31] F. Su and K. Chakrabarty. 2006. Benchmarks for digital microfluidic biochip design and synthesis. <https://ece.duke.edu/fs/Benchmark.pdf>
- [32] Tsun-Ming Tseng, Bing Li, Tsung-Yi Ho, and Ulf Schlichtmann. 2015. Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 141:1–141:6. <https://doi.org/10.1145/2744769.2744899>
- [33] T. M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T. Y. Ho, I. E. Araci, and U. Schlichtmann. 2017. Columba 2.0: A Co-Layout Synthesis Tool for Continuous-Flow Microfluidic Biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2017), 1–1. <https://doi.org/10.1109/TCAD.2017.2760628>
- [34] T. M. Tseng, M. Li, B. Li, T. Y. Ho, and U. Schlichtmann. 2016. Columba: Co-layout synthesis for continuous-flow microfluidic biochips. In *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. <https://doi.org/10.1145/2897937.2897997>
- [35] Marc A Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, and Stephen R Quake. 2000. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science* 288, 5463 (2000), 113–116.
- [36] David I. Walsh, David S. Kong, Shashi K. Murthy, and Peter A. Carr. 2017. Enabling Microfluidics: from Clean Rooms to Makerspaces. *Trends in Biotechnology* 35, 5 (May 2017), 383–392. <https://doi.org/10.1016/j.tibtech.2017.01.001>
- [37] Qin Wang, Yizhong Ru, H. Yao, T. Y. Ho, and Y. Cai. 2016. Sequence-pair-based placement and routing for flow-based microfluidic biochips. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 587–592. <https://doi.org/10.1109/ASP-DAC.2016.7428075>
- [38] Q. Wang, H. Zou, H. Yao, T. Y. Ho, R. Wille, and Y. Cai. 2018. Physical Co-Design of Flow and Control Layers for Flow-Based Microfluidic Biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 6 (June 2018), 1157–1170. <https://doi.org/10.1109/TCAD.2017.2748003>
- [39] Angela R. Wu, Joseph B. Hiatt, Rong Lu, Joanne L. Attema, Neethan A. Lobo, Irving L. Weissman, Michael F. Clarke, and Stephen R. Quake. 2009. Automated microfluidic chromatin immunoprecipitation from 2,000 cells. *Lab on a Chip* 9, 10 (May 2009), 1365–1370. <https://doi.org/10.1039/B819648F>
- [40] Hailong Yao, Tsung-Yi Ho, and Yici Cai. 2015. PACOR : Practical Control-Layer Routing Flow with Length-Matching Constraint for Flow-Based Microfluidic Biochips. In *Design Automation Conference*.