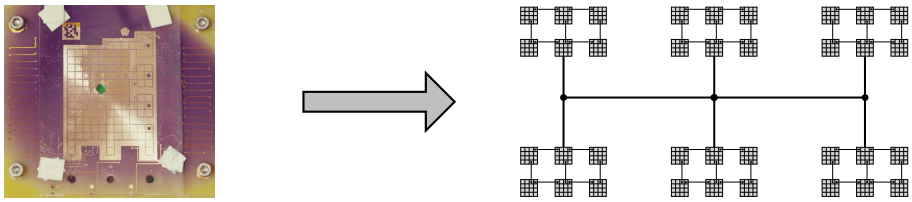
Mega-Microfluidcs

Scaling Up Laboratory Automation using Commodity Devices

Max Willsey
University of Washington

Luis Ceze
University of Washington



Microfluidic automation promises to make biology and chemistry more precise and efficient. Wetlabs are already using various technologies to automate part of their workflows.

To scale things even further, various companies offer cloud lab services. In this paradigm, the user submits a job (and perhaps mails in some reagents), and robots in a warehouse perform the necessary fluidic tasks. Finally, the result is mailed back to the user, or in some cases just the relevant data is sent.

Our work [10] in ASPLOS this year explores a more dynamic approach to microfluidic automation, raising the possibility of running more dynamic protocols that "close the loop" on automated experimentation. But that work only looked at the hardware and software necessary to run protocols on one device, a device that is prohibitively small for many workloads. That said, those hardware design decisions have big benefits: the device is cheap, easy to use, and modular. Inspired by modern cloud-scale computer systems, this WACI presentation will propose *cloud-scale, dynamic, multi-tenant microfluidic automation powered by small, cheap commodity components* .

Scaling Up the Hardware Digital microfluidic (DMF) platforms tend to be small, on the scale of hundreds of electrodes. This is due to complications in manufacturing: typically every electrode requires independent control, and sourcing components that operate at the necessary voltage levels is difficult. If this technology catches on, mass production could change this, but it's still unlikely that single devices will satisfy large numbers of complex protocols at the same time.

Instead, we propose cloud-scale microfluidic automation composed of many smaller, cheaper devices. This will require some kind of fluidic interconnect, the details of which remain very much unclear. In our ASPLOS paper, we used small peristaltic pumps to perform input/output between the DMF board and a test tube of reagents. We anticipate that the same technique can be used to facilitate board-to-board communication at the cost of one pump per connection.

To minimize the number of needed connections, we picture a hierarchical topology, where clusters of DMF devices are densely connected to one another, and those clusters are more sparsely connected to one another. The top-level connection could use a different pumping technology, possibly including valves allowing it to operate more generally as a bus instead of a direct connection. All of this poses a challenge to current routing techniques, which focus on uniform topologies (like a grid of electrodes).

Isolation & Virtualization We envision providing large-scale microfluidic automation as a *dynamic* service to users. Instead of submitting static jobs as graphs, we want users to write rich, complex programs whose fluidic portions (which may not be known ahead of time) are executed on the microfluidic platform concurrently.

Multi-tenant use calls for some mechanism of isolation. Our current ASPLOS paper begins to address this by handing out opaque droplet ids and abtracting away location. This prevents one user's program from clobbering another, but does not prevent one protocol from starving another. Cloud computing platforms approach this by providing virtual processors, and limiting dynamic resource usage to the preallocated number of processors. We picture something similar for microfluidics: we provide a virtual DMF device of a certain size[1], and the runtime system limits a user to using that much space (although it may not correspond 1:1 to any physical DMF device in the system).

Fault Tolerance Individual electrodes on a single DMF device are prone to failure, and we addressed this in our ASPLOS paper with a computer vision based error-correction system. However, this does nothing for you if many electrodes fail, essentially rendering a whole DMF device useless. In a setting with many devices though, we could attempt to migrate the protocol to a different physical DMF device(s) in a way that's transparent to the user.

[1] And with a certain number of peripherals (DNA sequencers, heaters, etc.)

This WACI submission was accompanied by a short video. Below, we have included some references to works in microfluidics, especially those we think will be relevant to the ASPLOS community. More references and a more thorough discussion of related work can be found in our ASPLOS paper about a programming system for a single microfluidic device.

References

[1] Mirela Alistar and Urs Gaudenz. 2017. OpenDrop: An Integrated Do-It-Yourself Platform for Personal Use of Biochips. *Bioengineering* 4, 2 (2017), 45.

[2] Ahmed M Amin, Mithuna Thottethodi, TN Vijaykumar, Steven Wereley, and Stephen C Jacobson. 2007. Aquacore: a programmable architecture for microfluidics. In *ACM SIGARCH Computer Architecture News*, Vol. 35. ACM, 254–265.

[3] Vaishnavi Ananthanarayanan and William Thies. 2010. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of Biological Engineering* 4, 1 (2010), 13.

[4] Kihwan Choi, Alphonsus H.C. Ng, Ryan Fobel, and Aaron R. Wheeler. 2012. Digital Microfluidics. *Annual Review of Analytical Chemistry* 5, 1 (2012), 413–440. https://doi.org/10.1146/annurev-anchem-062011-143028

[5] Christopher Curtis, Daniel Grissom, and Philip Brisk. 2018. A compiler for cyber-physical digital microfluidic biochips. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. ACM, 365–377.

[6] Daniel Grissom, Christopher Curtis, Skyler Windh, Calvin Phung, Navin Kumar, Zachary Zimmerman, O'Neal Kenneth, Jeffrey McDaniel, Nick Liao, and Philip Brisk. 2015. An open-source compiler and PCB synthesis tool for digital microfluidic biochips. *INTEGRATION, the VLSI journal* 51 (2015), 169–193.

[7] Jason Ott, Tyson Loveless, Chris Curtis, Mohsen Lesani, and Philip Brisk. 2018. BioScript: programming safe chemistry on laboratories-on-a-chip. *Proceedings of the ACM on Programming Languages* 2, OOPSLA (2018), 128.

[8] Synthace. 2018. Antha. https://synthace.com/introducing-antha

[9] Transcriptic. 2018. Autoprotocol. http://autoprotocol.org/

[10] Max Willsey, Ashley P. Stephenson, Chris Takahashi, Pranav Vaid, Bichlien H. Nguyen, Michal Piszczek, Christine Betts, Sharon Newman, Sarang Joshi, Karin Strauss, and Luis Ceze. 2019. Puddle: A Dynamic, Error-Correcting, Full-Stack Microfluidics Platform. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. ACM, New York, NY, USA. https://doi.org/10.1145/3297858.3304027