EC500 D1 - Computational Synthetic Biology For Engineers

Homework 3
Spring 2017

April 3, 2017

BioElectronic Design

Due: 05/03/17

Overview

Goal: Form an understanding of the complexity of microfluidic design problems and how tools can be used to solve them.

Activities:

1. **(75 Points)** Find a published continuous-flow based microfluidic device and clearly report its function. Your report must include the following attributes about your device:
   - Dimensions
   - Parameters
   - Valve control sequences
   - Experimental protocol
   - Number of citations and a brief description of its use in one citing article
   - Fabrication technique
2. **(75 Points)** Use 3DμF to create the device. Deliverables include:
   - SVG file
   - JSON file
   - STL file
   - Description of features that could not be drawn using 3DμF
3. **(75 Points)** Use Neptune to automatically place and route your chosen device.
   (a) Create a Mint description of the device
   (b) Use Neptune to place and route the MINT description. Deliverables include:
      - SVG design output of Fluigi
      - Init_Parameters.ini file
   (c) Fabricate the Device

Note: Grading will be done on the deliverables so please ensure that all the required deliverables are included to ensure proper grading.

Activity #1

In this activity you will do a literature search for an interesting microfluidic device. The device **must** be based on continuous flow (not digital microfluidics, droplet-based microfluidics, acoustophoresis, electrophoresis, etc.). Your device should have valves that are pneumatically actuated. The following journals are a great place to begin your search for an interesting device. This by no means an exhaustive list:

- Lab on a Chip
- Microfluidics and Nanofluidics
- Biomedical Microdevices

In order to make the best decision regarding which device to analyze for this assignment, please read the entire homework assignment before selecting a device. This is to ensure you understand what you will be doing with this device (ex., if you pick a device without valves, it will be impossible to complete many of the activities in this assignment). Points will be awarded based on the quality of the deliverables as well as how interesting and complex the device is (an extremely simple device that does not lend itself well to designing with tools will have a hard time earning top marks in this assignment).

The deliverable for Activity #1 is a report that **clearly** describes the following:

- The device's physical dimensions
- What are some of the device's parameters (channel width, number of inputs, number of valves, etc.)? How does the function of the device change as these parameters change?
- How does this device scale? Can you tile the device to make larger devices? Cascade the device and do intermediate operations?
  - Example: The device in Figure 1 of this homework is essentially a grid of specialized valves, so scaling it would require adding a row or column to the grid, which would increase the number of valves required. A complete answer for this section of the report could include a formula for how the device scales (i.e., relate valves to rows/columns mathematically).
- A clear description of the experimental protocol your device was designed to perform (ex., DNA assembly, cell sorting, etc.).
- A temporal description of the valve control sequence.
  - Hint: You should clearly link the physical architecture of the device to the experimental protocol that the device was designed to perform. All intermediate steps performed on the microfludic chip should be clearly documented here.

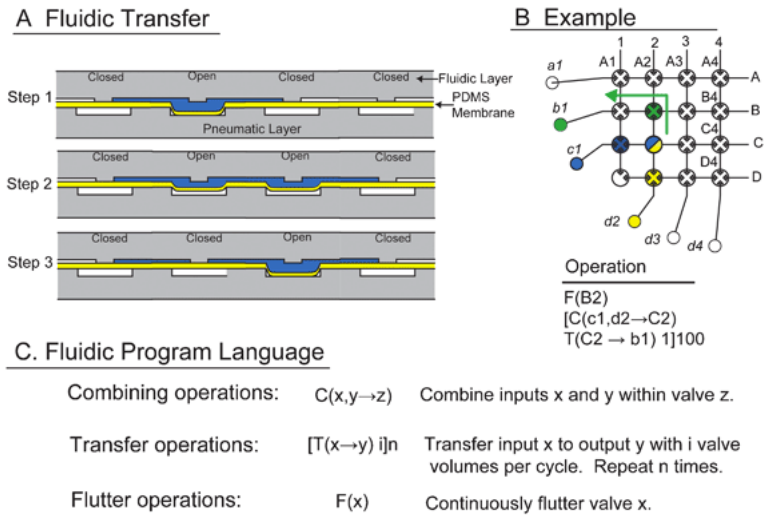

Fig. 1 (A) Cross sectional view of a microvalve array showing the steps for transfer of fluids between microvalves. (B) Schematic of a combining operation showing labeling of valve array and inputs. Inputs from wells *c*1 and *d*2 are drawn in *via* valves C1 and D2, combined in valve C2, and transferred to output well *b*1, with a total fluidic transfer of one valve volume per cycle. The cycle is then repeated 100 times. (C) Fluidic program language. Combining operations indicate inputs (x,y…) followed by the microvalve or microvalves where combining occurs (z). Transfer operations indicate the microvalve or storage well input (x) followed by the output (y), and the number of microvalve volumes transferred per cycle (i). Operations within brackets are repeated *n* times. Flutter operations are performed continuously and indicate the microvalve fluttered (x).

Figure 1:

- A description of the ubiquity of the device. How many times has this device been cited? Choose a citing article and describe the context in which the device was used.
- A description of the fabrication technique used to create the device. Popular techniques are soft lithography, CNC milling, wet etching, etc.

Activity #2

3DμF is a user-friendly CAD tool for placing microfluidic components on a grid and exporting the design into a few useful file formats. Your job in Activity #2 is to use 3DμF to design the device chosen in Activity #1 and export the device in three file formats: JSON (to send to microfluidic control software), SVG (for use with a CNC mill), and STL (for 3D printing).

It is very likely that 3DμF will not have all of the features you need to properly design your device. Your job will be to clearly describe features that 3DμF cannot draw and create primitives for each feature that 3DμF does not support. The standard for defining a primitive is shown in the Mint Github Wiki, which you will use extensively in Activity #3.

Deliverables for Activity #2:

- SVG File
- JSON File
- STL File
- Description of features that could not be drawn using 3DμF, including primitive definitions for each feature.
  - Significant bonus points will be awarded for implementing the feature in javascript and submitting a pull request to the 3DμF Github repository

Activity #3

Activity Tasks:

1. Create a description of the of the device using Mint, a microfluidic hardware description language.
2. Include comments into the Mint description file.
3. Place and route the device using Neptune.

Deliverables

1. Mint Description of the microfludic Device named eg. 'device.uf'.
2. Clearly document the features that could not be implemented using Mint (PDF) showing images and the alternative components that were used in place of the original feature.
3. Fluigi input parameters used for routing named eg. 'fluigi.ini'.
4. Brief report on the changes incorporated in the PAR/DRC parameters for the device(PDF).
5. (Bonus) EPS design snapshots of the failed Designs (if any).
6. EPS design file of the generated design.
7. Log generated by Neptune.
8. Fabricated Device (Maximum weightage is given to the fabrication output).

Dealing with Fluigi/Mint Error

Since Fluigi is a research tool in active development, there might be some constructs or scenarios where the application will not work as expected or might not accurately report the errors in the Mint file.

The binary version of Fluigi that will be distributed will not have some of the actively developed functionality, namely the verilog device generation and the hierarchical design parsing (i.e. the 'IMPORT device' option).

In many cases the Place and Route (PAR) might fail because it does not find a good solution or finds a solution that would violate the Design Rule Check (DRC). In these event's the report should include all the failure messages and the snapshots generated by Fluigi to avail the bonus points and to demonstrate the limitations of the active research tool.

In the scenarios where Neptune crashes or fails, Github issues should created for each of the scenarios on the homework/Fluigi Github repository. This way the questions can be cleared quickly and the binaries can be patched to fix the errors. In addition to that if faced with any other difficulty please contact sanka@bu.edu requesting assistance/office hours.

Mint Tutorial

Mint is a microfluidic hardware description language used for describing microfluidic circuits and devices. Just like any other Hardware Description Language like Verilog or VHDL, Mint aims to help microfluidic designers leverage the use of CAD tools by allowing them to describe complex architectures in a human and machine readable format. By using Mint descriptions of the hardware. The designers can easily share and tweak the circuit and component parameters and speed up the entire experimental process.

An example of Mint Syntax can be seen below:

Mint FAQ

Q. How can I find the Mint documentation?

A. Mint's references can be found at Mint Github Wiki and Chapter 4 of Huang Thesis.

Q. How do I create and structure a Mint file and are there any examples?

A. The basic rules and example mint files can be seen at Netlist Rules and Examples

Q. What are the various primitives and modules available in Mint?

```
DEVICE test01

LAYER FLOW
PORT p1, p2, p3 r=500;
NODE n1;
H LONG CELL TRAP ct1 numChambers=10 chamberWidth=500
chamberLength=500 chamberSpacing=100 channelWidth=500;

CHANNEL c1 from p1 3 to n1 1 w=500;
CHANNEL c2 from p2 1 to n1 3 w=500;
CHANNEL c3 from n1 2 to ct1 1 w=500;
CHANNEL c4 from ct1 2 to p3 4 w=500;
END LAYER

LAYER CONTROL
PORT cp1, cp2 r=500;
VALVE v1 on c1 w=1000 l=500;
VALVE v2 on c2 w=1000 l=500;
CHANNEL c5 from cp1 2 to v1 4 w=500;
CHANNEL c6 from cp2 2 to v2 4 w=500;

END LAYER
```

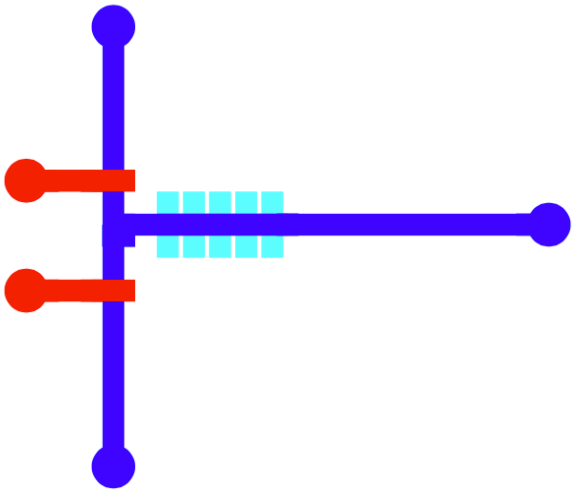

Figure 2: Netlist example. The netlist describes a simple device with three ports and a cell trap on the flow layer and two valves and ports on the control layer. The name of the device is given in the first line. The features and connections on the flow layer are described first, followed by those on the control layer. The design described in the netlist is shown on the right.

A. A complete list of primitives that can be used in in Mint can be found in the Primitives section of the documentation.

Q. What are the more complex module types available in Mint?

A. A complete list of predefined modules available in Mint can be found at Predefined Modules

Neptune Tutorial

Neptune is a specify design and build tool for Microfluidics. At its core, it contains a microfluidic place and route tool used for laying out the microfluidic components described in a Mint file and properly connecting the components according to the Mint description.

Neptune FAQ

Q. How do I get the Neptune binary?

A. Neptune can be found at Neptune Github Repo

The Fluigi (Core of Neptune) binary package can be found at EC500D1 Fluigi Repo. Download, place the binaries in your Homework repository and start with the homework.

Q. What are the dependencies for executing Fluigi?

A. Fluigi requires JRE 8 for it work. Please make sure that your system has the latest version of JRE installed and working in the command line.

This can be verified by typing the command in your command line :

```
$ java -version
```

Q. How do I run the Fluigi binary on a Mint file?

A. This can be seen at Usage

```
java [-Xmx????m] -jar fluigi mintfilename -i parameterFile
-o outputformat [-v|h]

Java Parameters:
-jar  : runs the jar file
[-Xmx????m] : Maximum memory that the application can use.

(???? specifies memory in megabytes)

Fluigi Parameters :
-i / --input : Input parameter file
-o / --output: output format
-v / --hdl : Verilog input (Will be disabled)
-h / --help : Prints out command line usage
```

Q. What do I do if my design fails to be generated by Fluigi?

A. First open an Github issue on the Neptune repository showing the error text, attaching an image of the design, Mint design file and the design parameters that have been changed from the default given. Second refer to the answer given in the question to try and mitigate the issue. **In case of unroutable designs, points will be awarded based on the interactions with the course staff recorded using the Github issues tracker.**

Q. How do I know if my device is unroutable or if Fluigi has broken?

A. Below is a table that captures the various reasons the PAR might fail and how to mitigate them.

Reason	Mitigation Strategy
1. The placer does not arrive at a good solution. 2. The placer arrives at a good solution but the DRC and other physical constraints of the design don't allow for successful routing.	1. By rerunning the design multiple times until the PAR gives a solution. 2. By trying to modify the design parameters, DRC constraints in the .ini file and fixing the positions of some of the components.

Q. Where can I find Fluigi Documentation, examples, etc. ?

A. The EC500D1/Fluigi Wiki is mirrored to the internal Fluigi Development Wiki and is the default documentation for running Fluigi.

Q. Where can I log the bugs for the BONUS POINTS ?

A. YES ! There are bonus points involved, you can log the issues on EC500D1/Fluigi Issues.

References

1. Jensen, E., Stockton, A., Chiesl, T., Kim, J., Bera, A., & Mathies, R. (2013). Digitally programmable microfluidic automaton for multiscale combinatorial mixing and sample processing. Lab on a Chip, 13(2), 288-296