Operation Placement for Application-Specific Digital Microfluidic Biochips

Mirela Alistar, Paul Pop, Jan Madsen
Technical University of Denmark, Lyngby
Biochips

Test tubes

- Automation
- Integration
- Miniaturization

Robotics

- Automation
- Integration
- Miniaturization

Microfluidics

- Automation
- Integration
- Miniaturization

Slide source: Krish Chakrabarty, Duke University
Droplet-based Biochips

Digital Microfluidic Biochips (DMBs)

Biochip from Duke University
Fluidic Operations

Video source: Advanced Liquid Logic http://www.liquid-logic.com/
• **General-Purpose Architecture**
  – Reconfigurable
  – Versatile
  – Fault-tolerant

• **Application-Specific Architecture**
  – Designed for one application
  – Reduced costs
    • Production costs
    • Reagent costs
Application-Specific Biochips

Biochip for Newborn Screening
http://www.liquid-logic.com/

Biochip for Sample Preparation
http://www.nugeninc.com/
Architecture Selection

- Reduced cost architectures
- Fault-tolerant architectures
- Increase the yield of DMBs

Architecture Selection Tool [ASPDAC, 2013]

Module Library
Component Library
Max No. of faults

Bioassay

Insulator degradation

Degradation of the electrode
Control electrodes (interdigitated design)
Biochemical Application Model

```
In S₁  In R₁  In S₂  In R₂  In S₂  In R₁  In S₃  In R₂
O₁     O₂     O₃     O₄     O₅     O₆     O₇     O₈
Mix  O₉  Mix  O₁₀  Mix  O₁₂  Mix  O₁₁
Merge  O₁₃  In B₁  O₁₄  In B₂  O₁₅  O₁₆  Merge
Dilution  O₁₇  Detection  O₁₉  Waste  O₂₀  O₂₁  Detection
Waste  O₂₂
```
Synthesis Flow

Biochemical Application → Synthesis
Fluidic Operations Library → Synthesis
Deadline requirement → Synthesis

Synthesis → Electrode actuation sequence → Application Execution

Computer

Biochip
Synthesis: Main steps

### Allocation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Area (cells)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>2 x 4</td>
<td>3</td>
</tr>
<tr>
<td>Mix</td>
<td>2 x 2</td>
<td>4</td>
</tr>
<tr>
<td>Dilution</td>
<td>2 x 4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Binding

![Binding diagram](source.png)

### Scheduling

- **Mixed O5**
- **Mixed O6**
- **Mixed O7**

![Scheduling diagram](source.png)
Synthesis: Main steps

### Allocation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Area (cells)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>2 x 4</td>
<td>3</td>
</tr>
<tr>
<td>Mix</td>
<td>2 x 2</td>
<td>4</td>
</tr>
<tr>
<td>Dilution</td>
<td>2 x 4</td>
<td>4</td>
</tr>
</tbody>
</table>

### Binding

- O1 → O6
- O2 → O3
- O3 → O5
- O5 → O4
- O5 → sink
- O1 → O7
- O4 → O7

### Scheduling

- M1
  - O5
  - O6
- M2
  - O7
Circular-route module

- M2 (CRM2) - 16 cycles
- Detector
- Waste

- CRM3 (8 cycles)
- M1 (CRM1) - 31 cycles
Circular-route module
Problem

• **Given**
  – Biochemical application
  – Application-specific architecture

• **Determine**
  – An circular-route placement of operations, so that the application completion time is minimized
CRM Library

Example of library for circular-route modules

<table>
<thead>
<tr>
<th>Operation</th>
<th>Circular-route module</th>
<th>Operation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>CRM$_1$</td>
<td>3.05</td>
</tr>
<tr>
<td>Mix</td>
<td>CRM$_2$</td>
<td>2.28</td>
</tr>
<tr>
<td>Mix</td>
<td>CRM$_3$</td>
<td>2.18</td>
</tr>
</tbody>
</table>
Building a CRM library

- Identify restricted rectangles (RRs)
- Use RRs as guidelines for obtaining CRMs
- CRMs are determined so that they minimize operation completion time
- CRMs are stored in a library

Determining circular-route modules
**Synthesis**

### Allocation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Circular-route module</th>
<th>Operation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>CRM$_1$</td>
<td>3.05</td>
</tr>
<tr>
<td>Mix</td>
<td>CRM$_2$</td>
<td>2.28</td>
</tr>
<tr>
<td>Mix</td>
<td>CRM$_3$</td>
<td>2.18</td>
</tr>
</tbody>
</table>

### Binding

![Binding Diagram]

### Placement

![Placement Diagram]

### Scheduling

![Scheduling Diagram]
Placement of CRMs

- Integrated with any available synthesis
- List Scheduling – based synthesis
- Select a CRM from library for each operation
  - such that the completion time is minimized
- Place the CRM on the biochip
- Schedule the operation
Experiments: setup

- **Biochemical applications:**
  - In-vitro diagnosis on human physiological fluids (IVD)
  - 3 synthetic benchmarks (SB$_1$ - SB$_3$)

- **Architecture:**
  - IVD – obtained with our Architecture Synthesis Tool

- **Implementation:**
  - Java

- **Evaluation:**
  - Efficiency in terms of application completion time
  - Comparison with Routing-based synthesis
Experimental results

<table>
<thead>
<tr>
<th>App. (ops*)</th>
<th>Arch.</th>
<th>MinR, MaxR</th>
<th>$\delta^R_G$ (s)</th>
<th>$\delta^{CRM}_G$ (s)</th>
<th>Deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVD (23)</td>
<td>45 (2, 2, 2)</td>
<td>[3, 5]</td>
<td>18.4</td>
<td>11.73</td>
<td>36</td>
</tr>
<tr>
<td>SB₁ (50)</td>
<td>96 (1, 2, 1)</td>
<td>[3, 5]</td>
<td>29.39</td>
<td>23.9</td>
<td>18.6</td>
</tr>
<tr>
<td>SB₂ (70)</td>
<td>103 (2, 2, 2)</td>
<td>[3, 6]</td>
<td>31.03</td>
<td>20.15</td>
<td>35</td>
</tr>
<tr>
<td>SB₃ (90)</td>
<td>125 (2, 2, 2)</td>
<td>[3, 8]</td>
<td>42.51</td>
<td>27.87</td>
<td>34</td>
</tr>
</tbody>
</table>

* We ignored the detection operations for experiments
IVD – in-vitro diagnosis, SB – synthetic benchmark
$\delta^R_G$ - application completion time
Conclusions

• Selection of architectures that minimize application completion time

• Strategy for placement of operations
  – We built a CRM-library
  – Better use of area
  – Better operation completion times

• Integration with our Architecture Selection tool
ListScheduling($Graph, \mathcal{C}, \mathcal{B}, \mathcal{P}$)

1. CriticalPath($Graph$)
2. repeat
3. $List = \text{GetReadyOperations}(Graph)$
4. $O_i = \text{RemoveOperation}(List)$
5. $t_i^{\text{start}} = \text{Schedule}(O_i, \mathcal{B}(O_i), \mathcal{C}, \mathcal{P})$
6. $t = \text{earliest time when a scheduled operation terminates}$
7. UpdateReadyList($Graph, t, List$)
8. until $List = \emptyset$
9. return $S$
Routing-based Operation Execution

<table>
<thead>
<tr>
<th>Operation</th>
<th>Area (cells)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>2 x 2</td>
<td>10</td>
</tr>
<tr>
<td>Mix</td>
<td>2 x 3</td>
<td>6</td>
</tr>
<tr>
<td>Mix</td>
<td>1 x 4</td>
<td>5</td>
</tr>
</tbody>
</table>

p^{90} = 0.1\%
p^{0} = 0.29\%
p^{00} = 0.58\%
p^{180} = -0.5\%

Droplet vs. Module Compilation

Module based
- module library
- black boxes
- protection borders

Droplet based
- routing base operation execution
- the position of the droplet is tracked
- better use of space

<table>
<thead>
<tr>
<th>Operation</th>
<th>Area (cells)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>2 x 4</td>
<td>3</td>
</tr>
<tr>
<td>Mix</td>
<td>2 x 2</td>
<td>4</td>
</tr>
<tr>
<td>Dilution</td>
<td>2 x 4</td>
<td>4</td>
</tr>
</tbody>
</table>
Module-based

Droplet-aware

Routing-based

Segregation border

2x4 Mixer

Deviated route

2x4 Mixer

Deviated route
Module-based

Segregation border

2x4 Mixer

Dispensing Reservoirs

Dis R

Dis B

Dis S

Routing – a post-synthesis step
### Droplet-aware

#### 2x4 Mixer

<table>
<thead>
<tr>
<th>Dis R</th>
<th>Dis B</th>
<th>Dis S</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Red droplet" /></td>
<td><img src="image2" alt="Green droplet" /></td>
<td><img src="image3" alt="Brown droplet" /></td>
</tr>
</tbody>
</table>

- Droplet position is known
- No segregation borders
- Better use of area
- Droplets stopped or deviated to avoid accidental merging
- Integrated routing

---

*Deviated route*
Routing-based

- No modules
- Contamination issues
Mixers

1x4 module, \( t = 4.6 \text{ s} \)

2x2 module, \( t = 9.95 \text{ s} \)
2x3 module, \( t = 6.1 \text{ s} \)
2x4 module, \( t = 2.9 \text{ s} \)

- flow reversibility issue
- one pivot issue

Mixing is faster if:
- No 180° moves
- Multiple pivots
- No change of direction

Routing-based operation execution
Optimization: Simulated Annealing

$A^0$ - initial architecture
$T^0$ - initial temperature
$T^L$ - temperature length
$\epsilon$ - cooling rate

$\text{temp} = T^0$

$A = A^0$

repeat

while (temp < $T^L$) do

$A_{\text{new}} = \text{moves}(A)$; //generate new architecture

$\delta = \text{Objective}(A) - \text{Objective}(A_{\text{best}})$;

if ($\delta < 0$)

$A_{\text{best}} = A_{\text{new}}$;

elseif (Math.random < $e^{-\delta/\text{temp}}$) //accept bad solutions with low probability

$A_{\text{best}} = A_{\text{new}}$;

endif

endwhile

$\text{temp} = \text{temp} \times \epsilon$

until stop criterion is true

$\text{Objective}(A) = \text{Cost}_A + W \times \max(0, \delta - D_G)$
ListScheduling(\(G, \mathcal{A}, \mathcal{L}\))

1. CriticalPath(\(G\))
2. repeat
3. \(List = GetReadyOperations(G)\)
4. \(O_i = RemoveOperation(List)\)
5. if Place(\(O_i, \mathcal{A}, \mathcal{L}\)) then
6. \(t_i^{start} = Schedule(O_i, \mathcal{A})\)
7. \(t = \) earliest time when an operation terminates
8. UpdateReadyList(\(G, t, List\))
9. end if
10. until \(List = \emptyset\)
11. return \(\delta_G\)
## Component Library

<table>
<thead>
<tr>
<th>Name</th>
<th>Unit cost</th>
<th>Dimensions (mm)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrode</td>
<td>1</td>
<td>1.5 × 1.5</td>
<td>N/A</td>
</tr>
<tr>
<td>Input Reservoir</td>
<td>3</td>
<td>1.5 × 4.5</td>
<td>2</td>
</tr>
<tr>
<td>Waste Reservoir</td>
<td>3</td>
<td>1.5 × 4.5</td>
<td>N/A</td>
</tr>
<tr>
<td>Capacitive Sensor</td>
<td>1</td>
<td>1.5 × 4.5</td>
<td>0</td>
</tr>
<tr>
<td>Optical Detector</td>
<td>9</td>
<td>4.5 × 4.5</td>
<td>8</td>
</tr>
</tbody>
</table>

## Virtual Devices Library

<table>
<thead>
<tr>
<th>Op.</th>
<th>Shape</th>
<th>Time (s) no faults</th>
<th>Time (s) (k = 1)</th>
<th>Time (s) (k = 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix</td>
<td>3 × 6</td>
<td>2.52</td>
<td>2.71</td>
<td>3.77</td>
</tr>
<tr>
<td>Mix</td>
<td>5 × 8</td>
<td>2.05</td>
<td>2.09</td>
<td>2.3</td>
</tr>
<tr>
<td>Mix</td>
<td>4 × 7</td>
<td>2.14</td>
<td>2.39</td>
<td>2.51</td>
</tr>
<tr>
<td>Mix</td>
<td>5 × 5</td>
<td>2.19</td>
<td>2.28</td>
<td>2.71</td>
</tr>
<tr>
<td>Mix</td>
<td>5 × 5 × 1</td>
<td>2.19</td>
<td>2.73</td>
<td>3.92</td>
</tr>
<tr>
<td>Mix</td>
<td>5 × 5 × 2</td>
<td>3.98</td>
<td>5.82</td>
<td>7.56</td>
</tr>
<tr>
<td>Dilution</td>
<td>3 × 6</td>
<td>4.4</td>
<td>4.67</td>
<td>4.11</td>
</tr>
<tr>
<td>Dilution</td>
<td>5 × 8</td>
<td>3.75</td>
<td>4.76</td>
<td>6.3</td>
</tr>
<tr>
<td>Dilution</td>
<td>4 × 7</td>
<td>3.88</td>
<td>4.22</td>
<td>4.46</td>
</tr>
<tr>
<td>Dilution</td>
<td>5 × 5</td>
<td>3.98</td>
<td>4.12</td>
<td>4.67</td>
</tr>
<tr>
<td>Split</td>
<td>1 × 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Storage</td>
<td>1 × 1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Build a CRM library

DetermineCRM(\(A, RR, MinR, MaxR, MinW, MaxW\))

1. \(L_{CRM} =\) List of circular route modules
2. \(FillArch(A, RR)\)
3. \(L_{SP} = GetStartPosition(A, RR, Radius)\)
4. for each \(StartPos\) in \(L_{SP}\) do
5.  
6.  for \(Radius\) from \(MinR\) to \(MaxR\) do
7.  
8.  for \(Window\) from \(MinW\) to \(MaxW\) do
9.  
10.  \(CRM =\) new circular route module
11.  repeat
12.   
13.   Next Pos = GetBestNeighbor(CRM, Radius, Window)
14.   InsertInRoute(CRM, Next Pos)
15.  
16. until Next Pos is StartPos
17.  UpdateList(\(L_{CRM}\), CRM)
18. end for
19. end for
20. end for
21. end for
22. InsertInList(\(L_{CRM}\), RR)
23. return \(L_{CRM}\)

Fig. 8: Determining circular-route modules