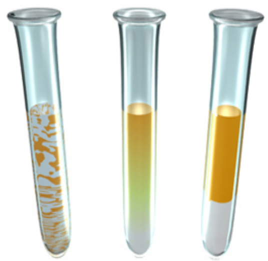
Operation Placement for Application-Specific Digital Microfluidic Biochips

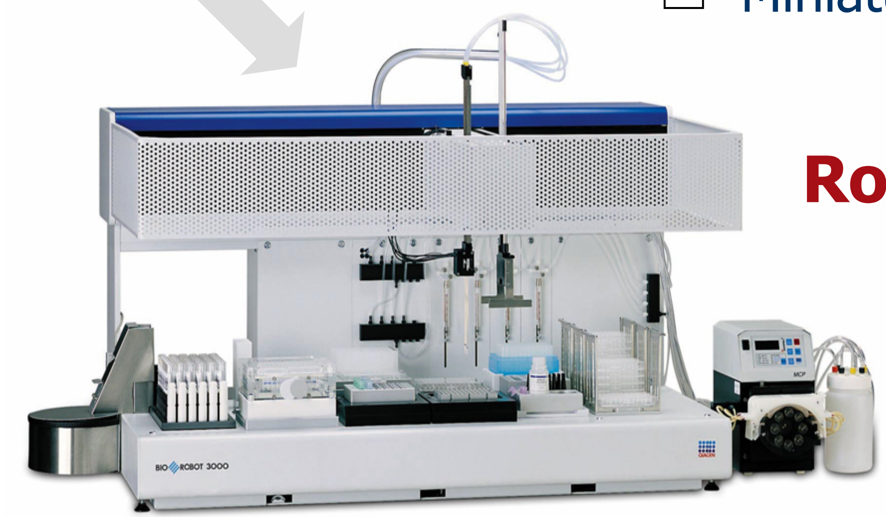
Mirela Alistar, Paul Pop, Jan Madsen
Technical University of Denmark, Lyngby

Biochips



Test tubes

- ☐ Automation
- ☐ Integration
- ☐ Miniaturization

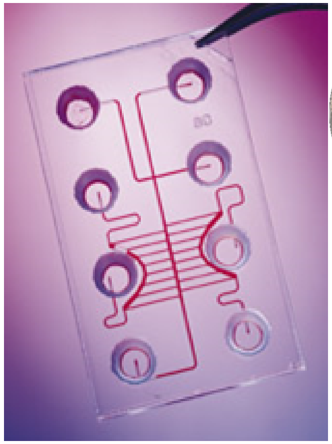


Robotics

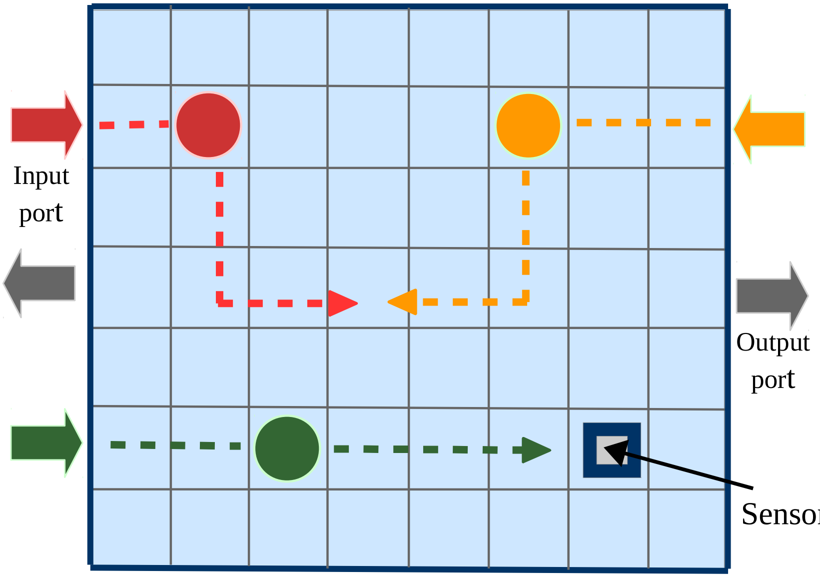
- ☑ Automation
- ☑ Integration
- ☐ Miniaturization

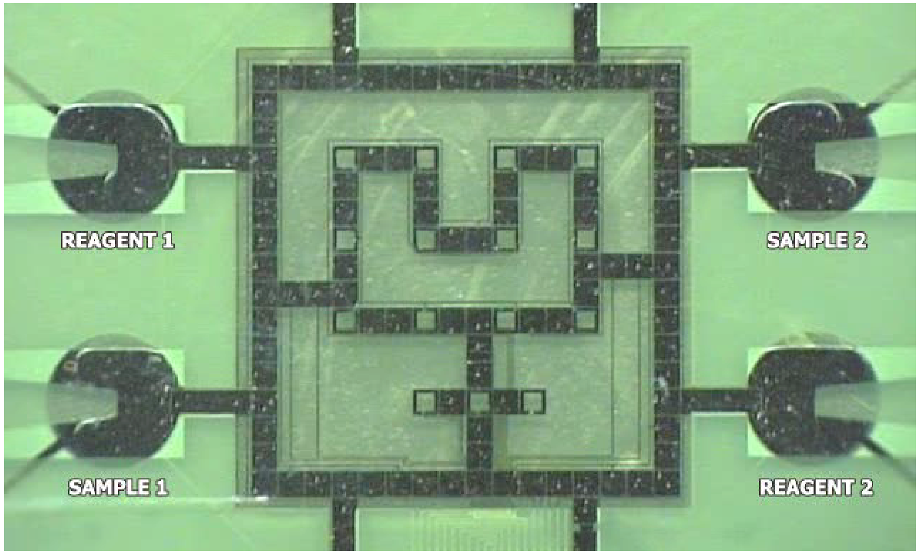
Microfluidics

- ☑ Automation
- ☑ Integration
- ☑ Miniaturization



Droplet-based Biochips

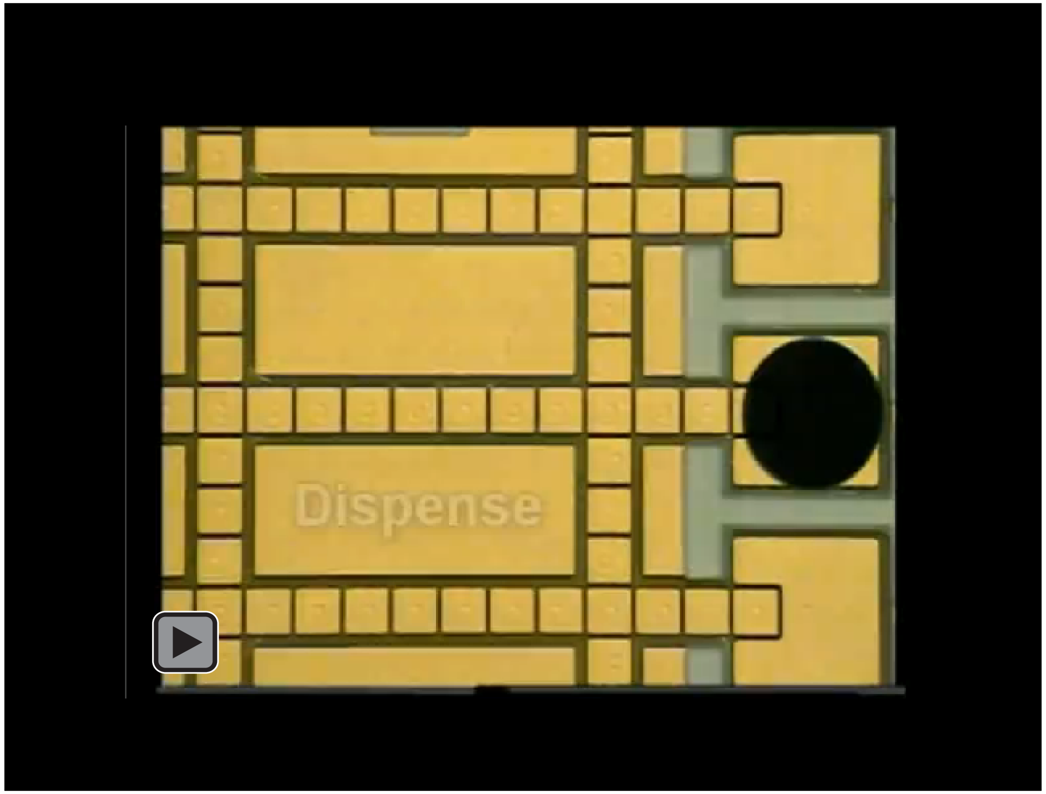




Biochip from Duke University

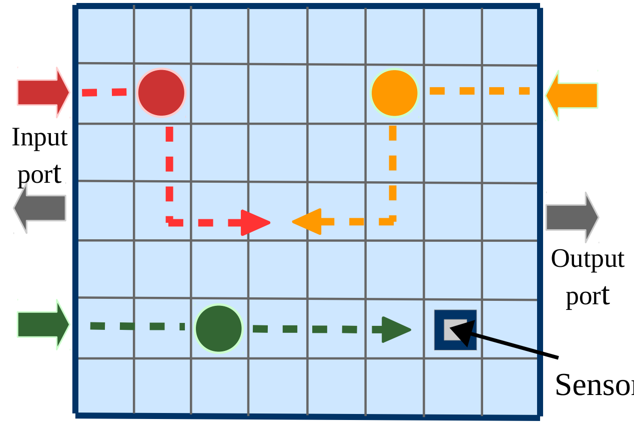
Digital Microfluidic Biochips (DMBs)

Fluidic Operations

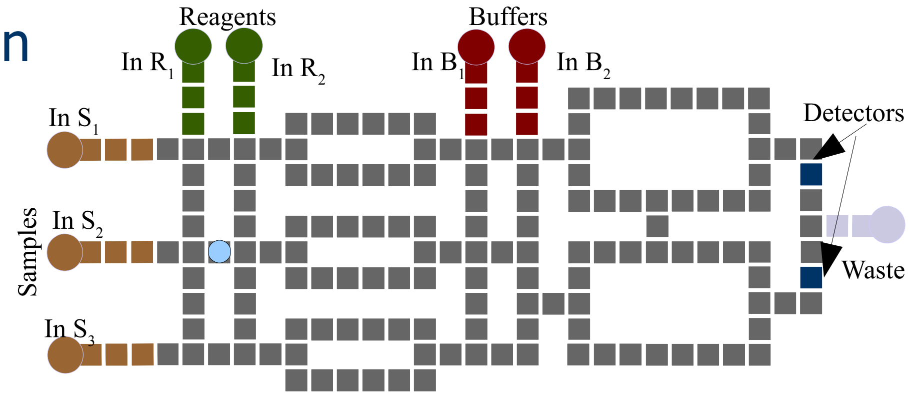


DMB Architecture

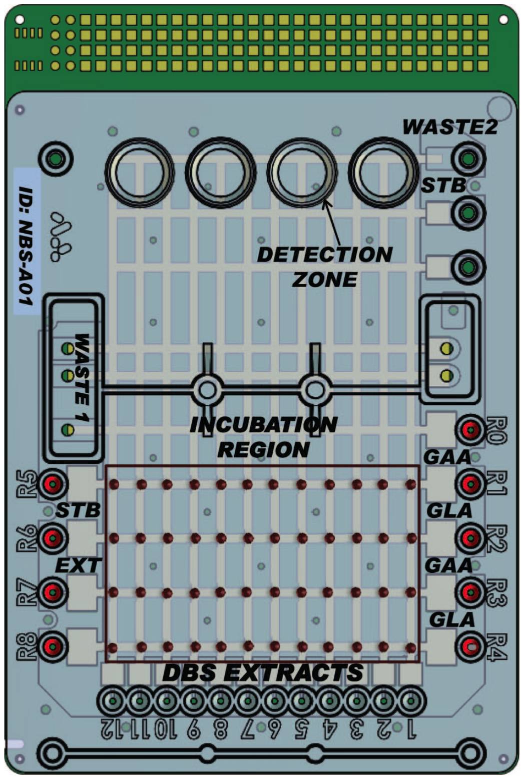
- General-Purpose Architecture
  - Reconfigurable
  - Versatile
  - Fault-tolerant



- Application-Specific Architecture
  - Designed for one application
  - Reduced costs
    - Production costs
    - Reagent costs

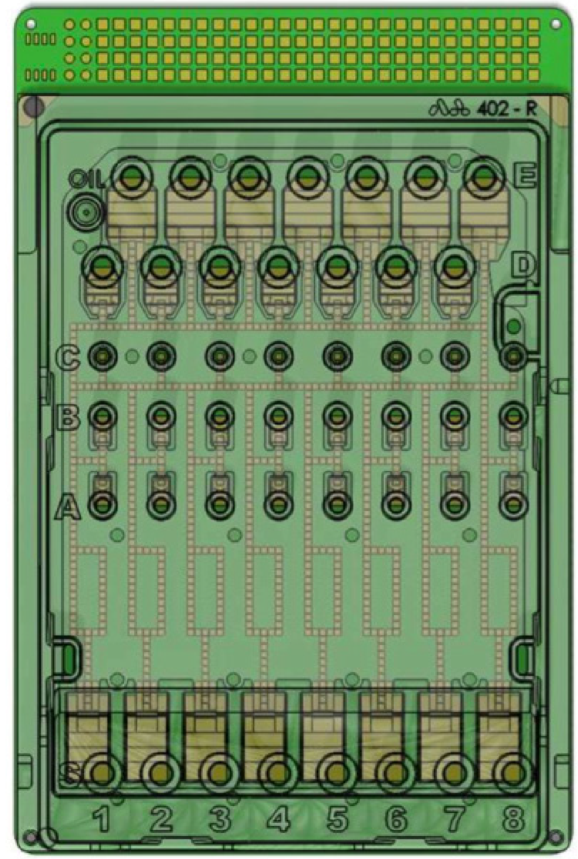


Application-Specific Biochips



Biochip for Newborn Screening

http://www.liquid-logic.com/

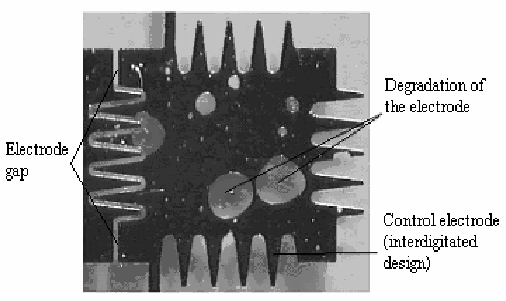


Biochip for Sample Preparation

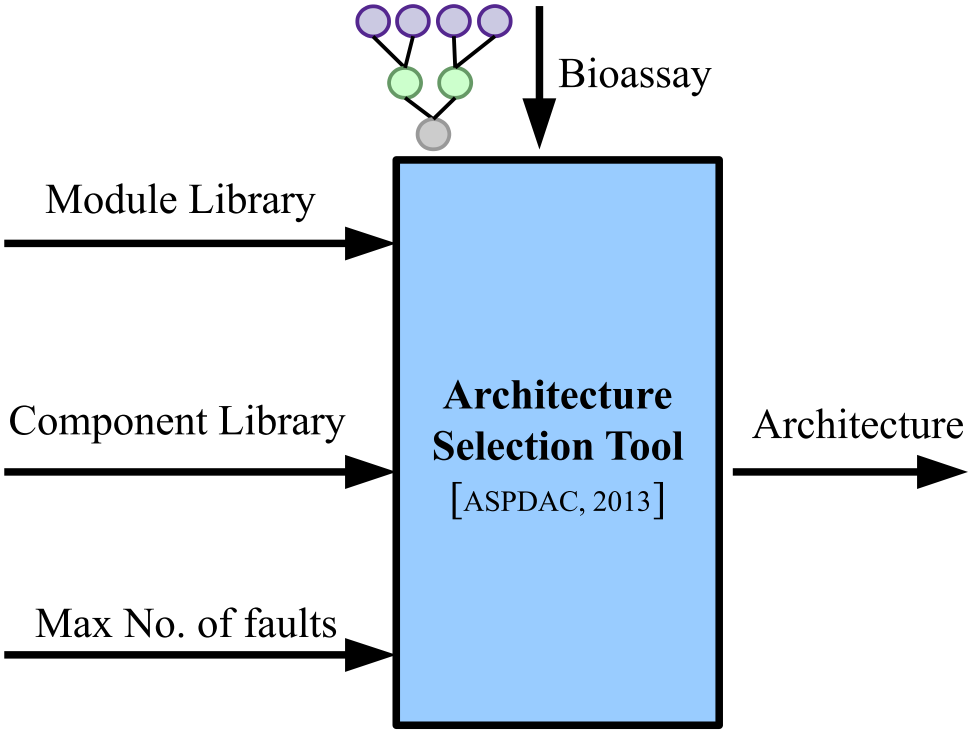
http://www.nugeninc.com/

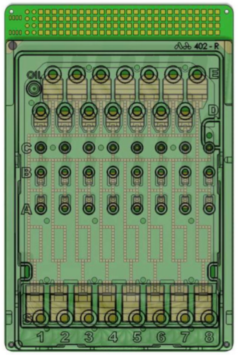
Architecture Selection

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

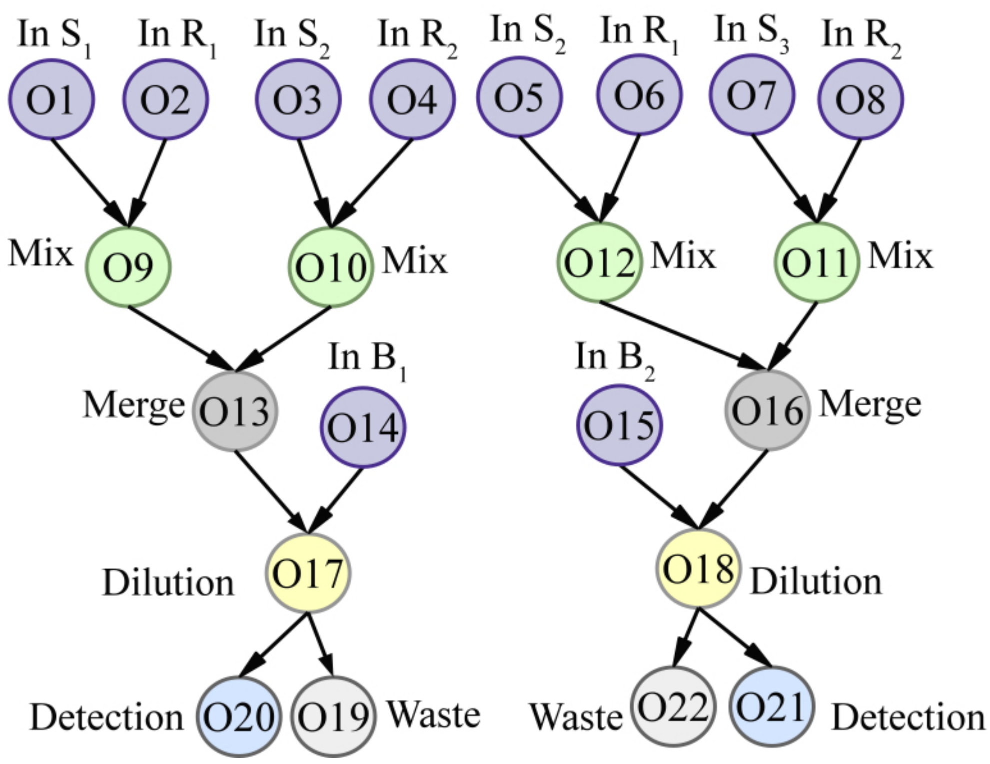


Insulator degradation

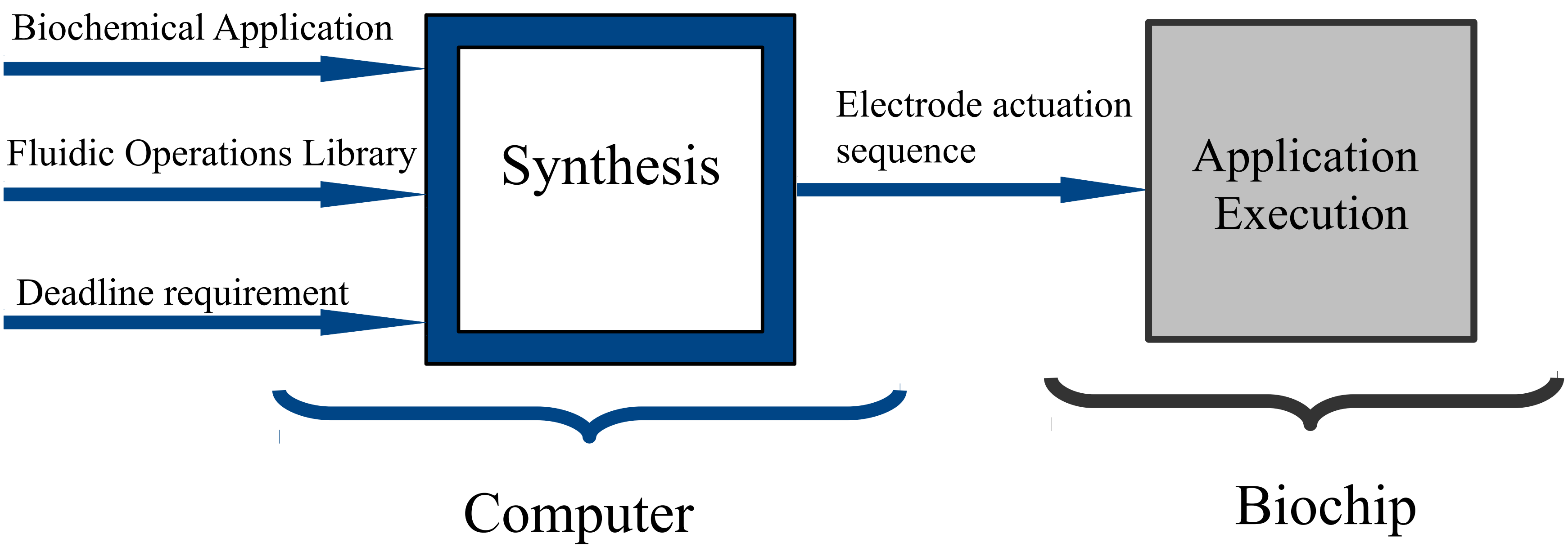




Biochemical Application Model



Synthesis Flow

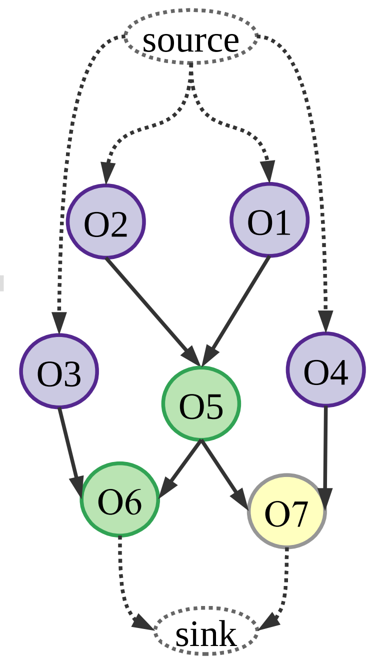


Synthesis: Main steps

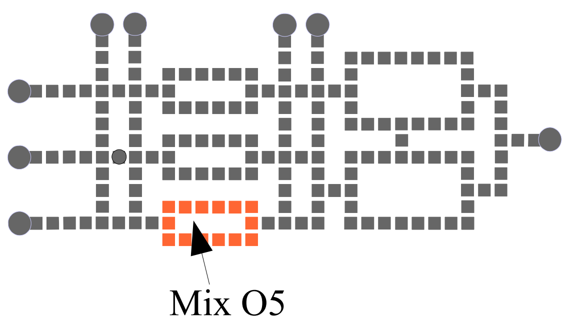
Allocation

Operation	Area (cells)	Time (s)
Mix	2 x 4	3
Mix	2 x 2	4
Dilution	2 x 4	4

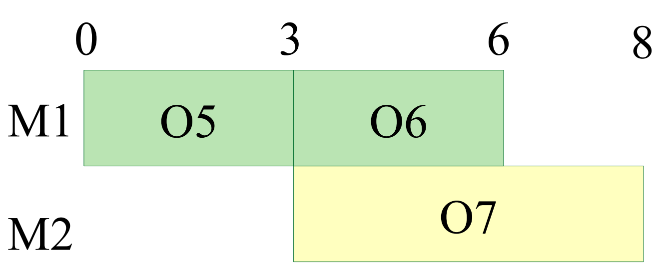
Binding



Placement



Scheduling

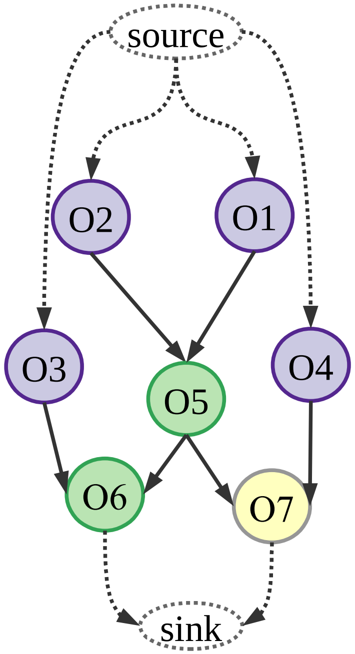


Synthesis: Main steps

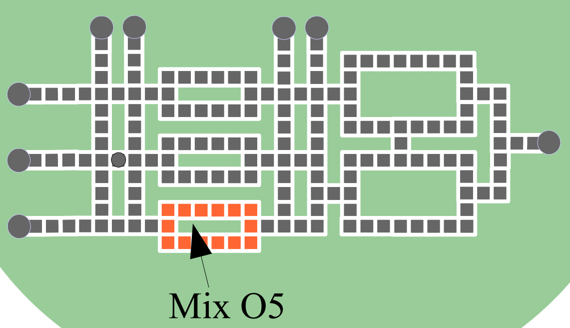
Allocation

Operation	Area (cells)	Time (s)
Mix	2 x 4	3
Mix	2 x 2	4
Dilution	2 x 4	4

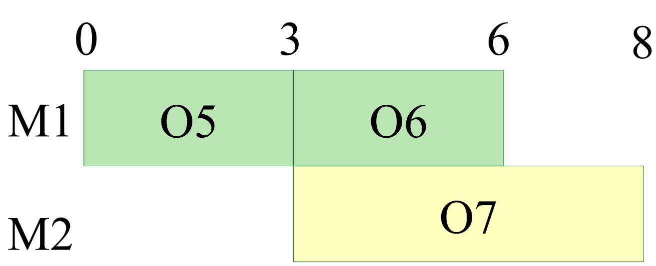
Binding



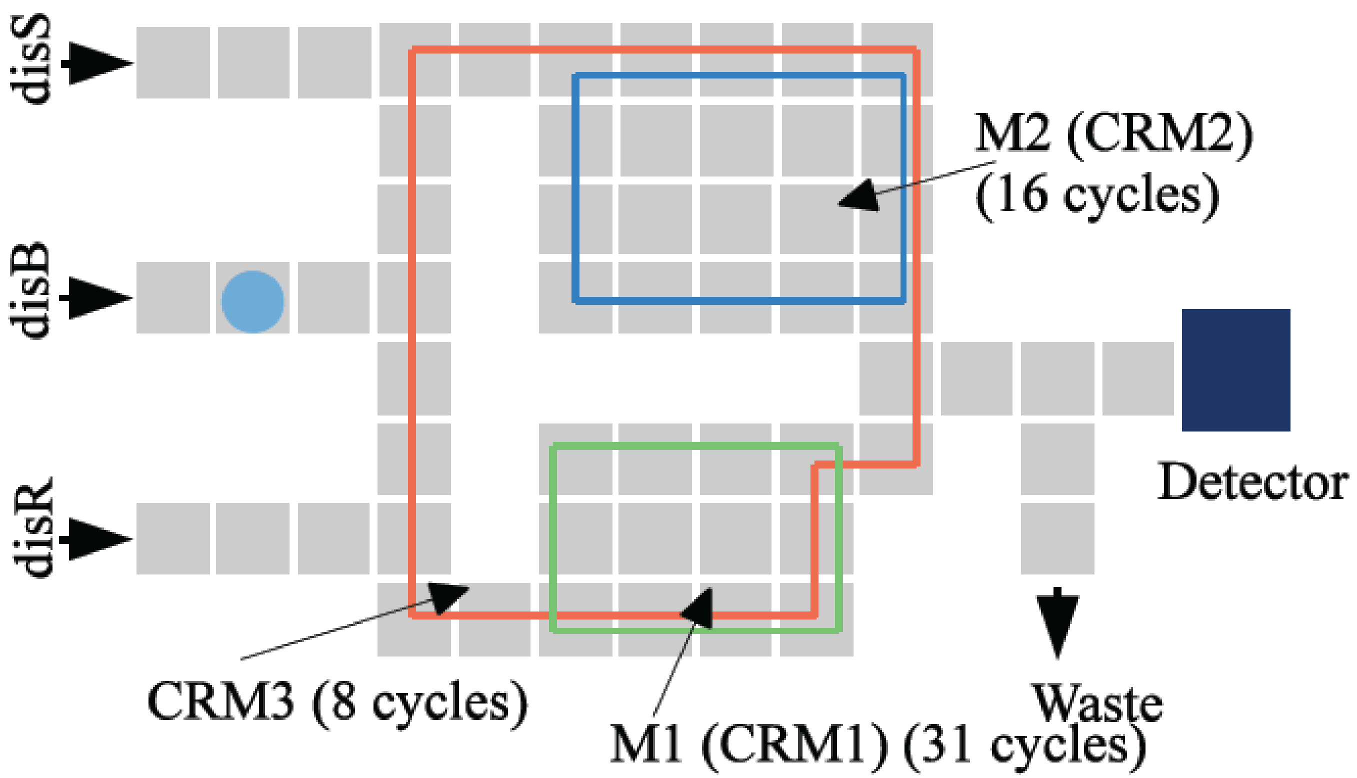
Placement



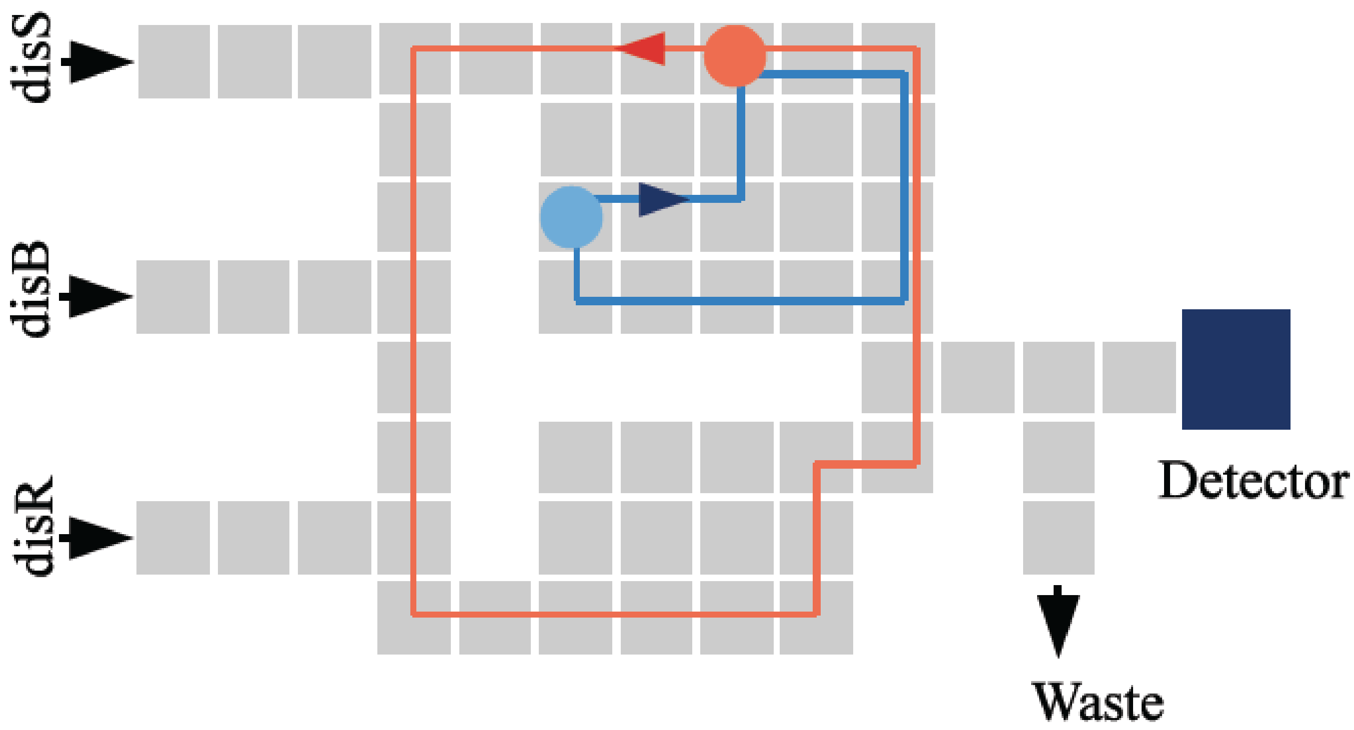
Scheduling



Circular-route module



Circular-route module

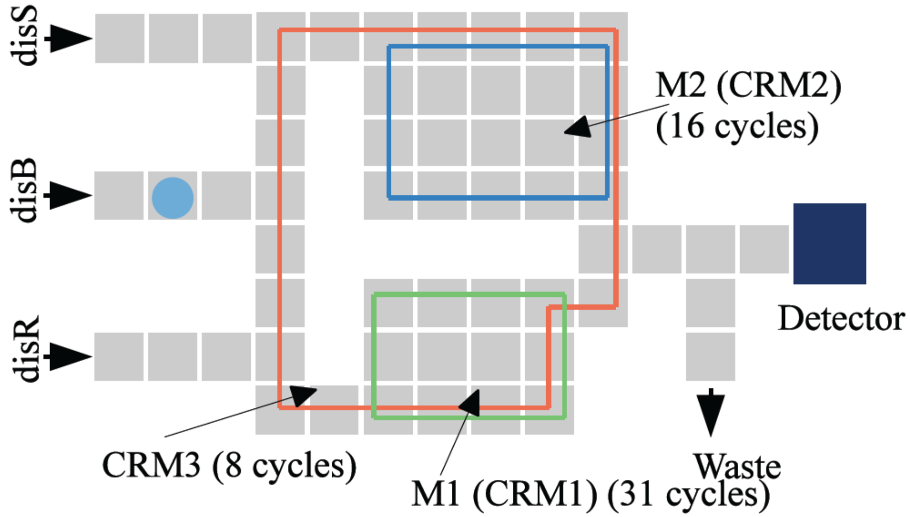


Problem

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

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

CRM Library

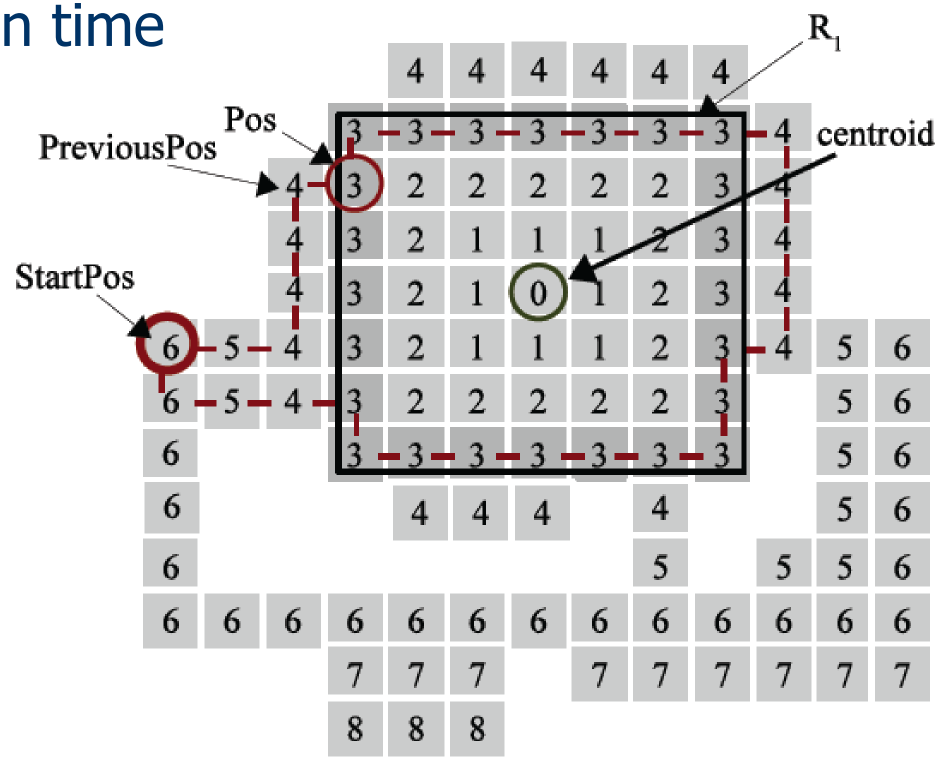


Example of library for circular-route modules

Operation	Circular-route module	Operation time (s)
Mix	CRM_1	3.05
Mix	CRM_2	2.28
Mix	CRM_3	2.18

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



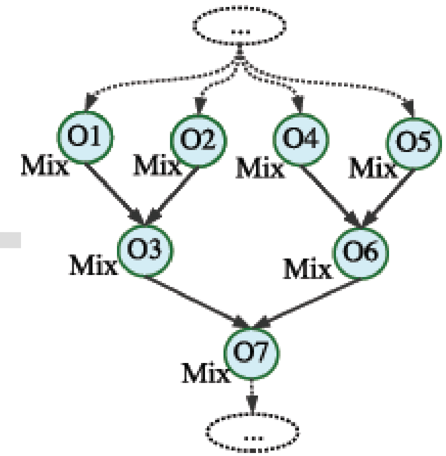
Determining circular-route modules

Synthesis

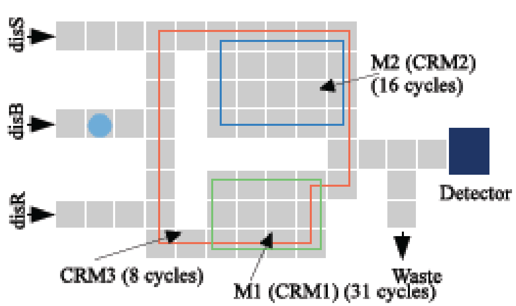
Allocation

Operation	Circular-route module	Operation time (s)
Mix	CRM_1	3.05
Mix	CRM_2	2.28
Mix	CRM_3	2.18

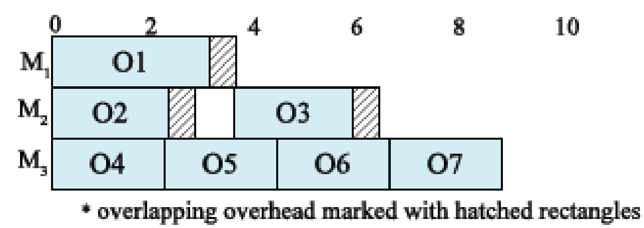
Binding



Placement



Scheduling



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

App. (ops*)	Arch.	*MinR, MaxR*	$\delta_{\mathcal{G}}^{R}$ (s)	$\delta_{\mathcal{G}}^{CRM}$ (s)	Deviation (%)
IVD (23)	45 (2, 2, 2)	[3, 5]	18.4	11.73	36
SB_1 (50)	96 (1, 2, 1)	[3, 5]	29.39	23.9	18.6
SB_2 (70)	103 (2, 2, 2)	[3, 6]	31.03	20.15	35
SB_3 (90)	125 (2, 2, 2)	[3, 8]	42.51	27.87	34

* We ignored the detection operations for experiments
IVD – in-vitro diagnosis, SB – synthetic benchmark
$\delta_{\mathcal{G}}$ - 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

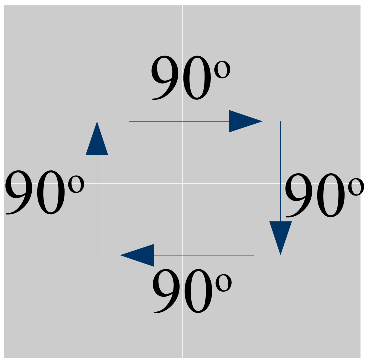
Backup slides

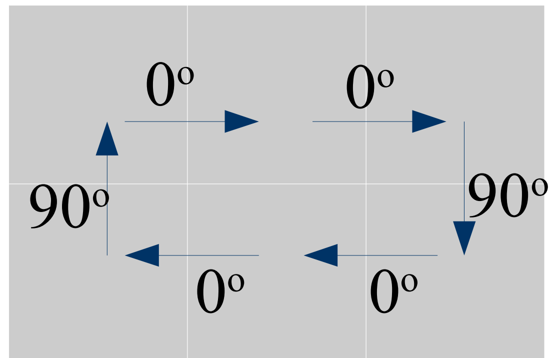
List Scheduling

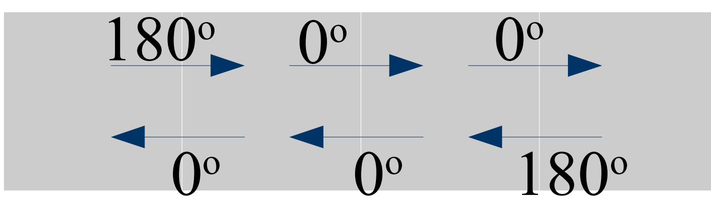
ListScheduling($Graph$, $\mathcal{C}$, $\mathcal{B}$, $\mathcal{P}$)

CriticalPath($Graph$)
repeat
    $List$ = GetReadyOperations($Graph$)
    O_i = RemoveOperation($List$)
    t_i^{start} = Schedule(O_i, $\mathcal{B}(O_i)$, $\mathcal{C}$, $\mathcal{P}$)
    t = earliest time when a scheduled operation terminates
    UpdateReadyList($Graph$, t, $List$)
until $List = \emptyset$
return $\mathcal{S}$

Routing-based Operation Execution







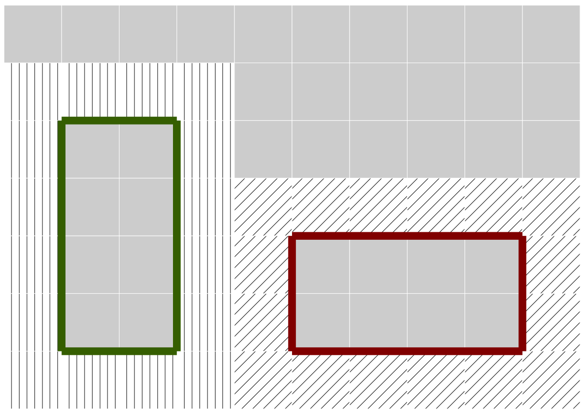
Operation	Area (cells)	Time (s)
Mix	2 x 2	10
Mix	2 x 3	6
Mix	1 x 4	5

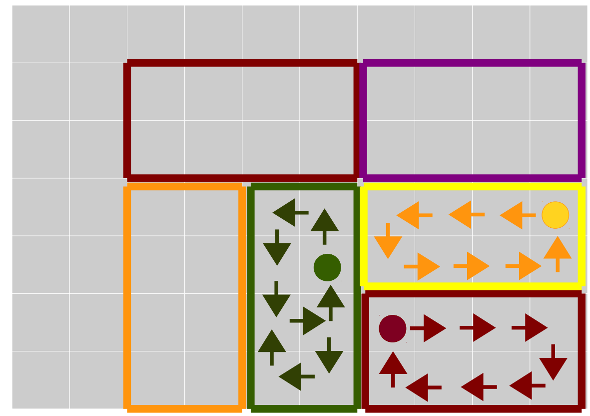
$p^{90} = 0.1\%$

$p^{0} = 0.29\%$ $p^{00} = 0.58\%$

$p^{180} = -0.5\%$

Droplet vs. Module Compilation





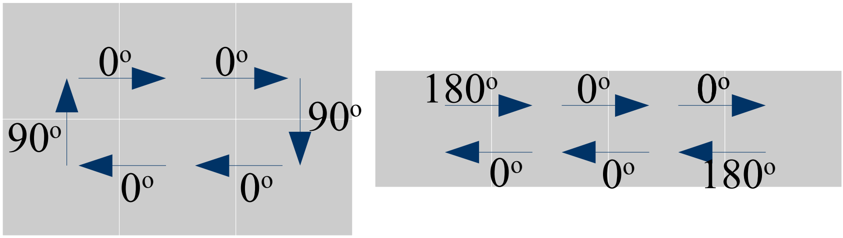
Module based

- module library
- black boxes
- protection borders

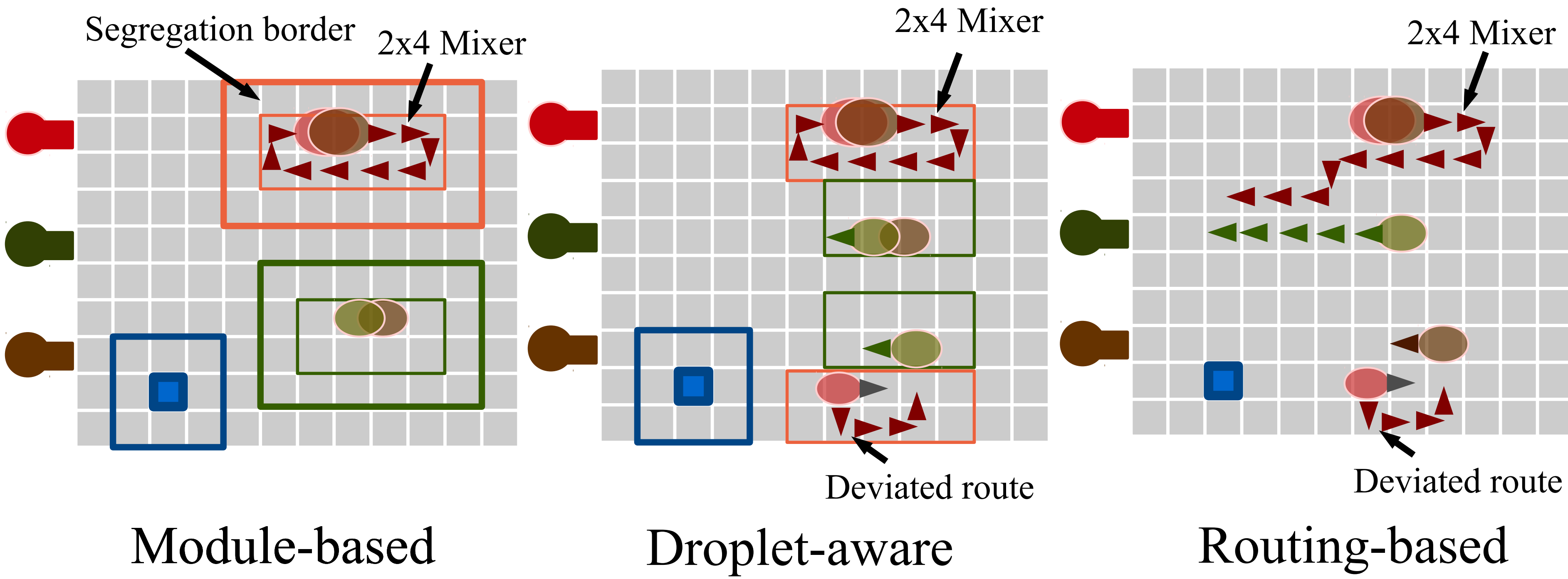
Droplet based

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

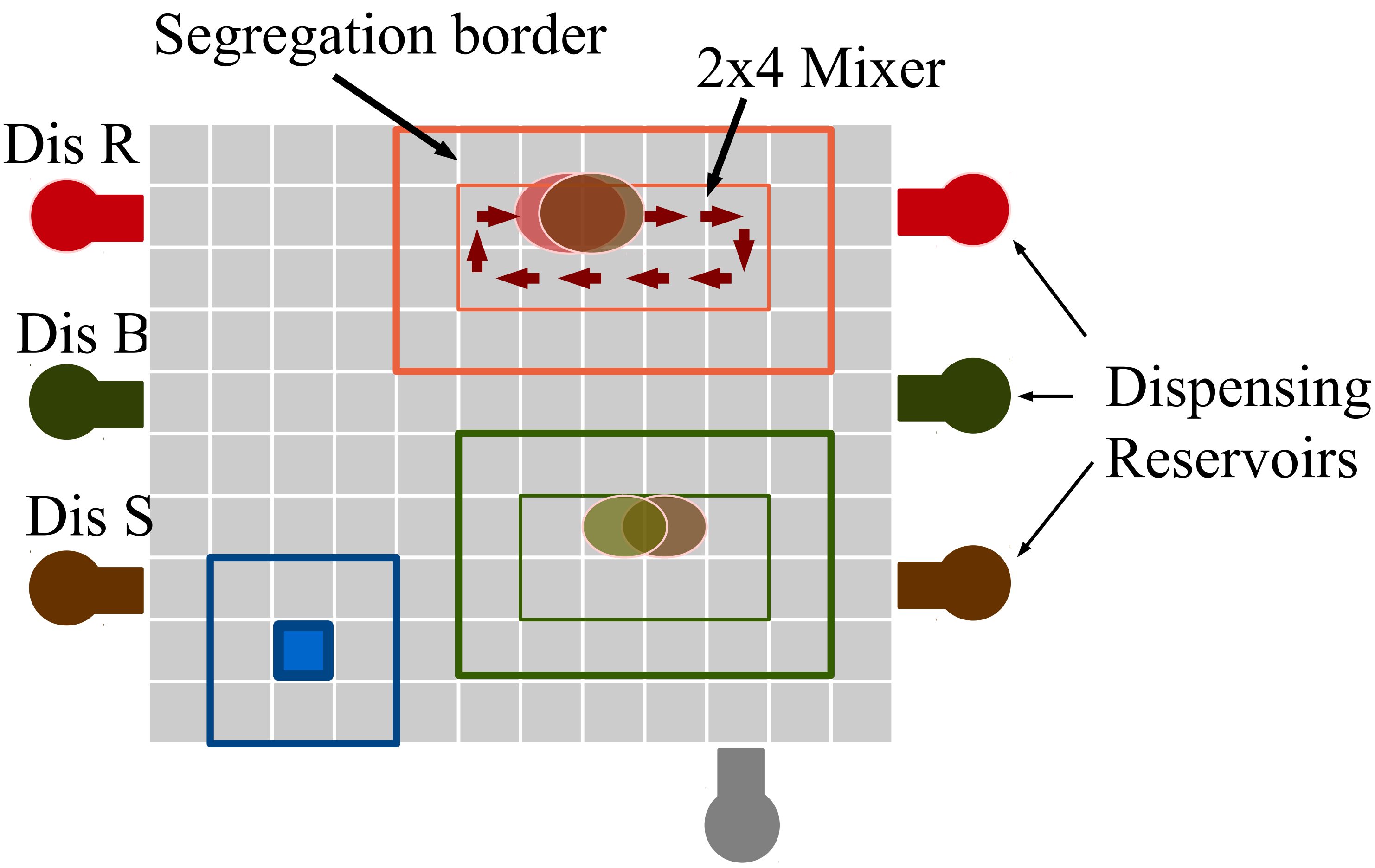
Operation	Area (cells)	Time (s)
Mix	2 x 4	3
Mix	2 x 2	4
Dilution	2 x 4	4



Placement

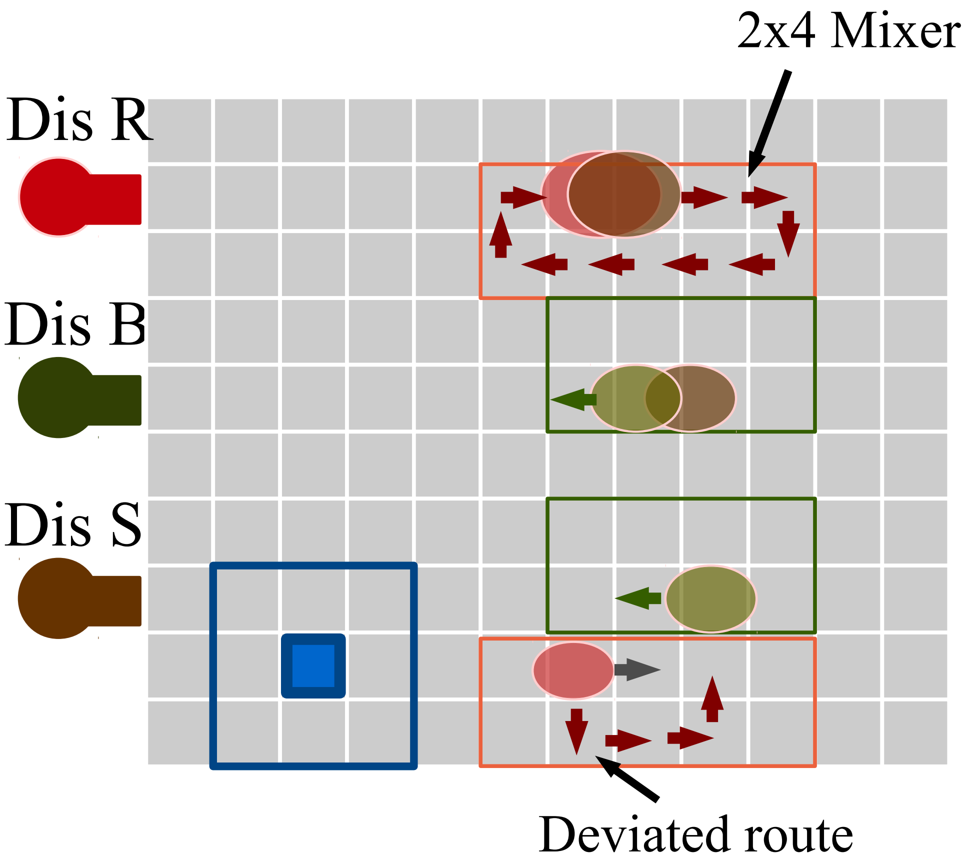

Segregation border
2x4 Mixer
2x4 Mixer
2x4 Mixer
Deviated route
Deviated route
Module-based
Droplet-aware
Routing-based

Module-based



Routing – a post-synthesis step

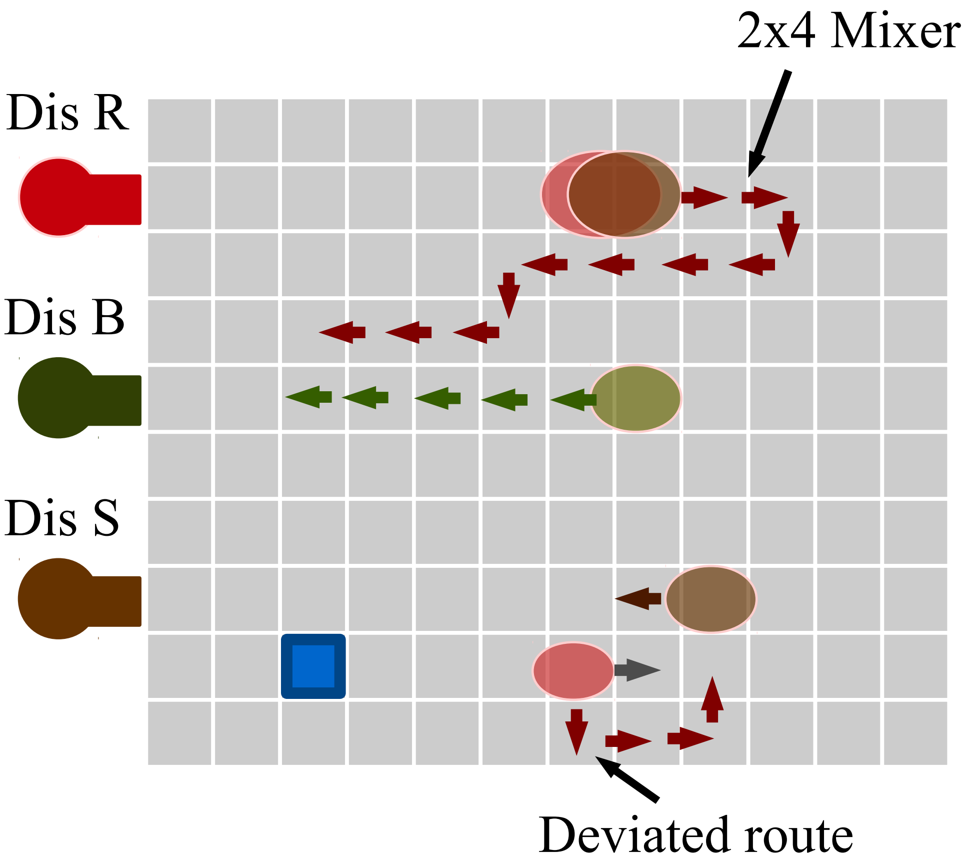
Droplet-aware



- Droplet position is known

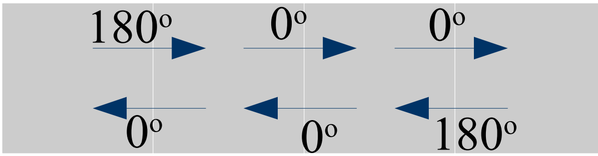
- No segregation borders
- Better use of area
- Droplets stopped or deviated to avoid accidental merging
- Integrated routing

Routing-based



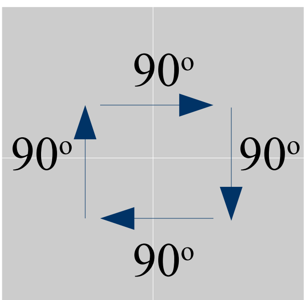
- No modules
- Contamination issues

Mixers



1x4 module, t = 4.6 s

- flow reversibility issue
- one pivot issue



2x2 module, t = 9.95 s

2x3 module, t = 6.1 s

2x4 module, t = 2.9 s

Mixing is faster if:

- No 180^o moves
- Multiple pivots
- No change of direction

Routing-based operation execution

Optimization: Simulated Annealing

$$Objective(\mathcal{A}) = Cost_{\mathcal{A}} + W \times max(0, \delta_{\mathcal{G}}^{k} - D_{\mathcal{G}})$$

```
𝒜⁰ - initial architecture
T⁰ - initial temperature
Tᴸ - temperature length
eps – cooling rate
temp = T⁰;
𝒜 = 𝒜⁰;
repeat
   while (temp < Tᴸ) do
       𝒜new = moves(𝒜); //generate new architecture
       delta = Objective(𝒜) - Objective(𝒜best);
       if (delta<0)
           𝒜best = 𝒜new;
       elseif (Math.random < e^(-delta/temp)) //accept bad solutions with low probability
           𝒜best = 𝒜new;
       endif
   endwhile
   temp = temp * eps;
until stop criterion is true
```

ListScheduling($\mathcal{G}$, $\mathcal{A}$, $\mathcal{L}$)

CriticalPath($\mathcal{G}$)
repeat
    $List$ = GetReadyOperations($\mathcal{G}$)
    O_i = RemoveOperation($List$)
    if Place($O_i, \mathcal{A}, \mathcal{L}$) **then**
        t_i^{start} = Schedule(O_i, $\mathcal{A}$)
        t = earliest time when an operation terminates
        UpdateReadyList($\mathcal{G}$, t, $List$)
    end if
until $List = \emptyset$
return $\delta_{\mathcal{G}}$

Component Library

Name	Unit cost	Dimensions (mm)	Time (s)
Electrode	1	1.5×1.5	N/A
Input Reservoir	3	1.5×4.5	2
Waste Reservoir	3	1.5×4.5	N/A
Capacitive Sensor	1	1.5×4.5	0
Optical Detector	9	4.5×4.5	8

Virtual Devices Library

Op.	Shape	Time (s) no faults	Time (s) $k=1$	Time (s) $k=2$
Mix	3×6	2.52	2.71	3.77
Mix	5×8	2.05	2.09	2.3
Mix	4×7	2.14	2.39	2.51
Mix	5×5	2.19	2.28	2.71
Mix	$5\times5\times1$	2.19	2.73	3.92
Mix	$5\times5\times2$	3.98	5.82	7.56
Dilution	3×6	4.4	4.67	4.11
Dilution	5×8	3.75	4.76	6.3
Dilution	4×7	3.88	4.22	4.46
Dilution	5×5	3.98	4.12	4.67
Split	1×1	0	0	0
Storage	1×1	N/A	N/A	N/A

Build a CRM library

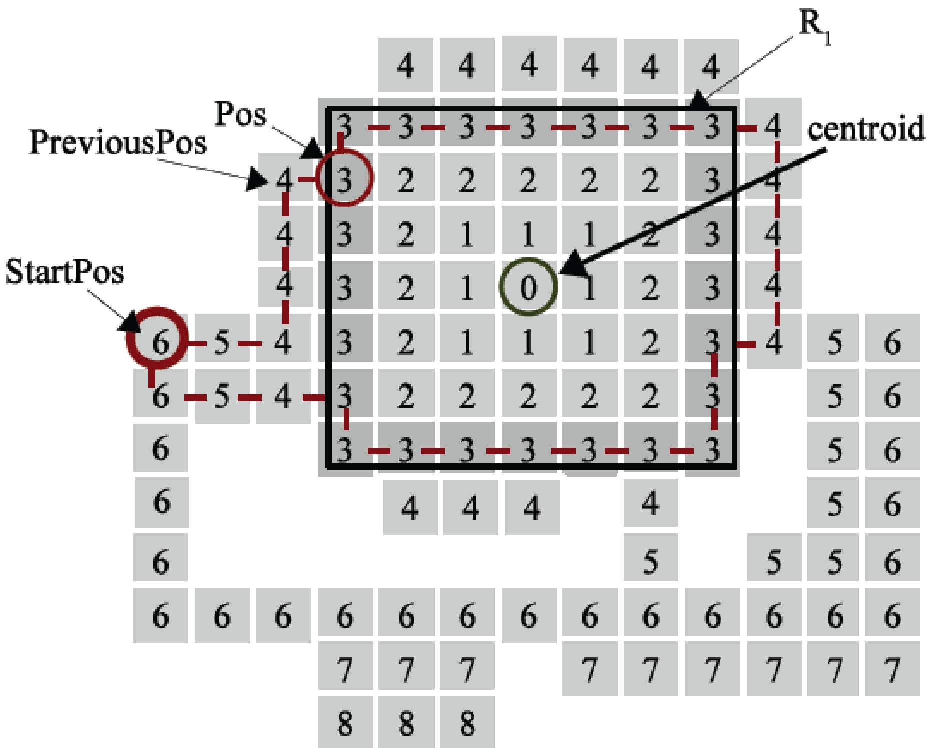


Fig. 8: Determining circular-route modules

DetermineCRM($\mathcal{A}, RR, MinR, MaxR, MinW, MaxW$)

```
L_CRM = List of circular route modules
FillArch(A, RR)
L_SP = GetStartPosition(A, RR, Radius)
for each StartPos in L_SP do
   for Radius from MinR to MaxR do
      for Window from MinW to MaxW do
         CRM = new circular route module
         repeat
            NextPos = GetBestNeighbor(CRM,Radius,Windo
            InsertInRoute(CRM,NextPos)
         until NextPos is StartPos
         UpdateList(L_CRM,CRM)
      end for
   end for
end for
InsertInList(L_CRM,RR)
return L_CRM
```