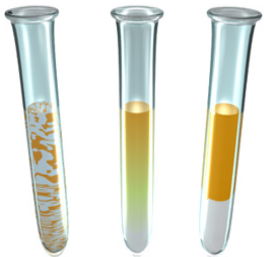


Application-Specific Fault-Tolerant Architecture Synthesis for Digital Microfluidic Biochips

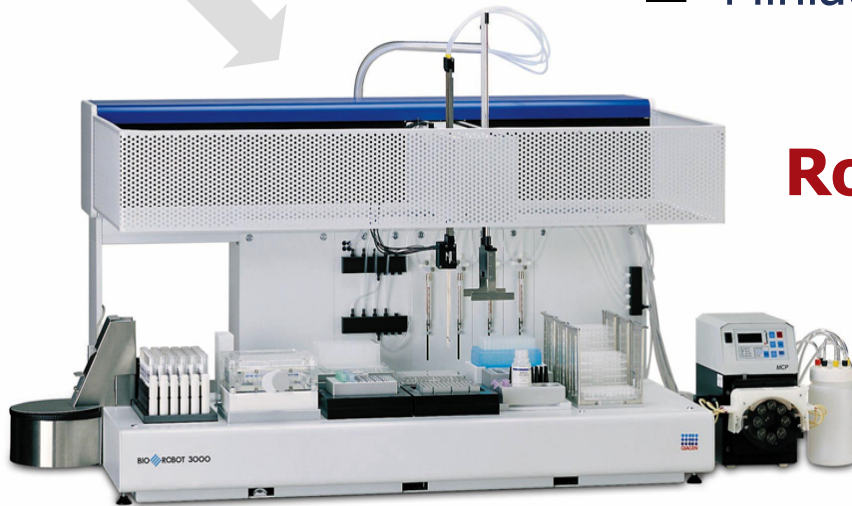
Mirela Alistar, Paul Pop, Jan Madsen
Technical University of Denmark, Lyngby





Test tubes

- Automation
- Integration
- Miniaturization

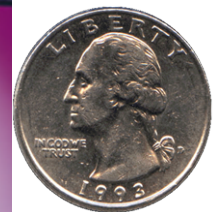
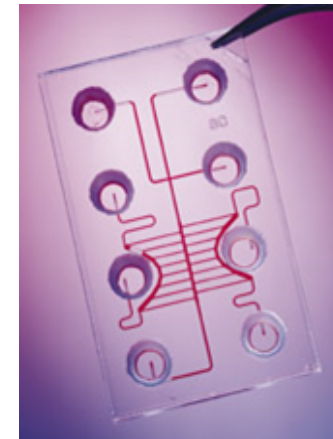


Robotics

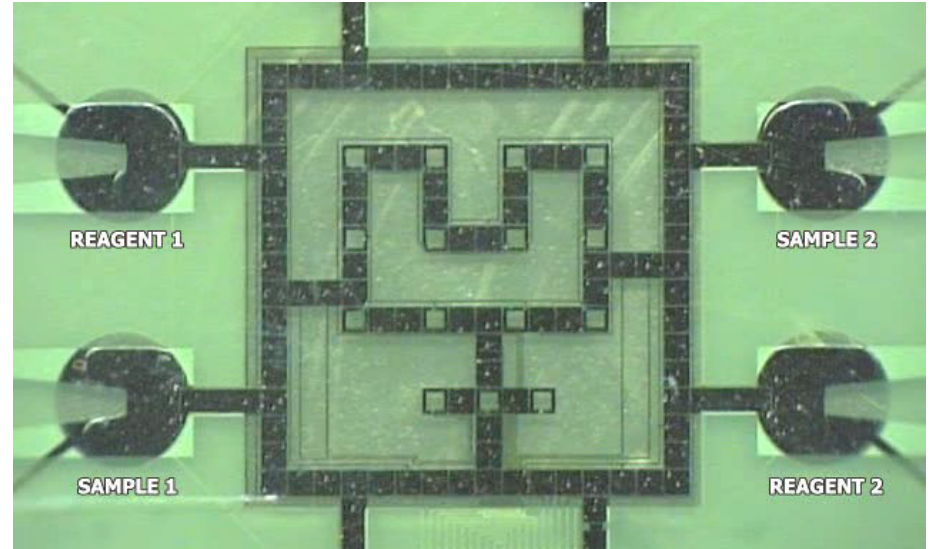
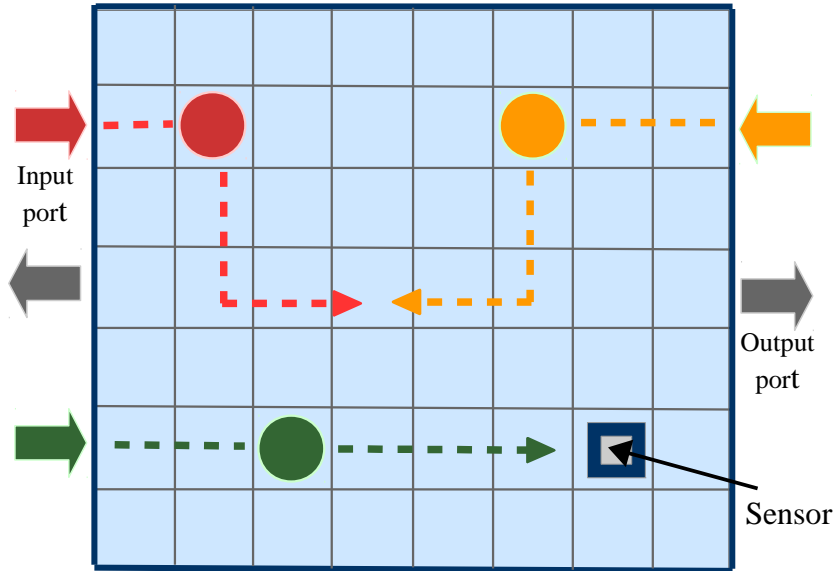
- Automation
- Integration
- Miniaturization

Microfluidics

- Automation
- Integration
- Miniaturization



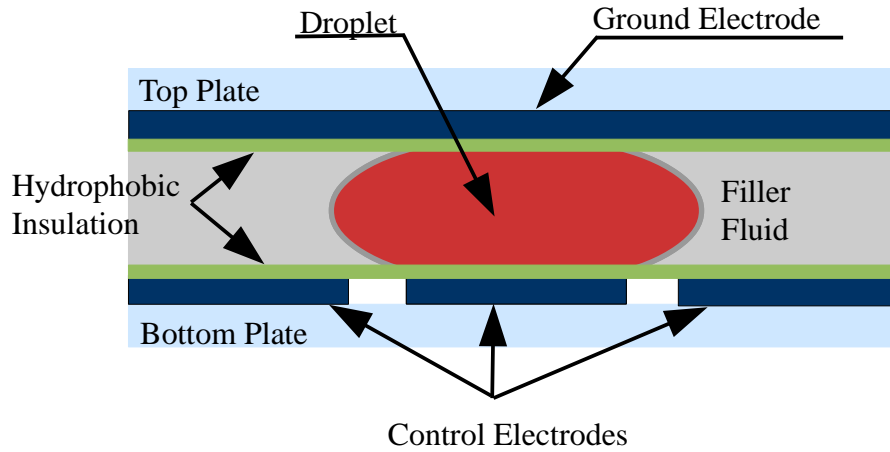
Droplet-based Biochips



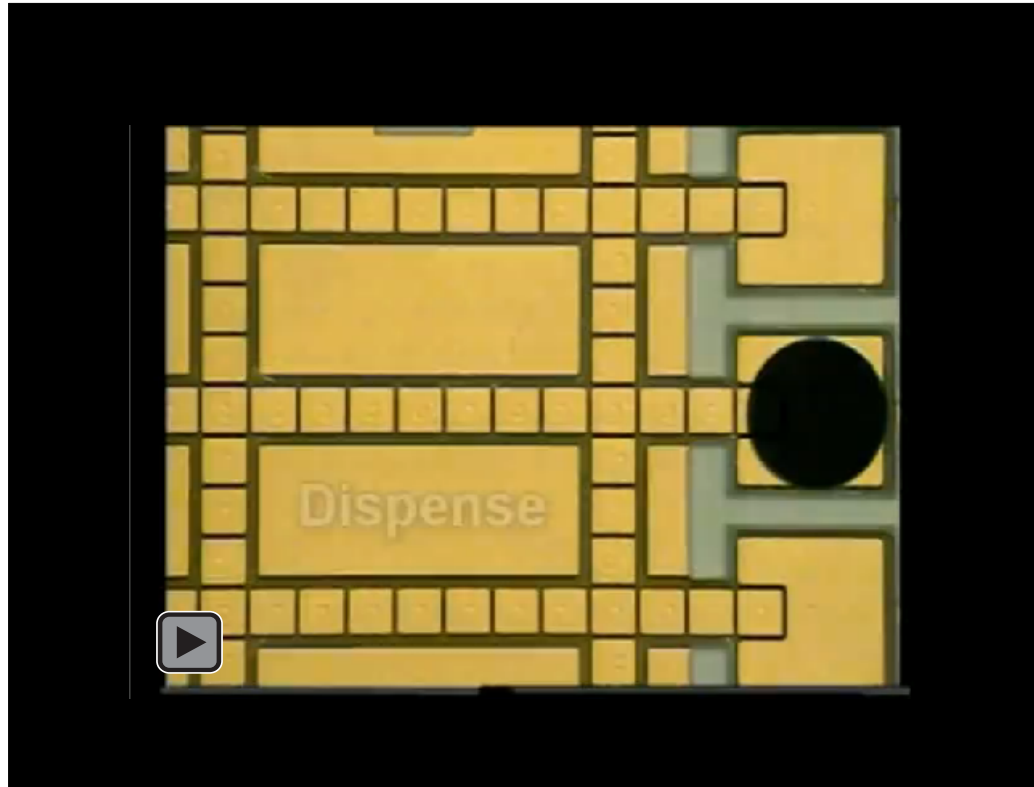
Biochip from Duke University

Digital Microfluidic Biochips (DMB)

Electrowetting on Dielectric



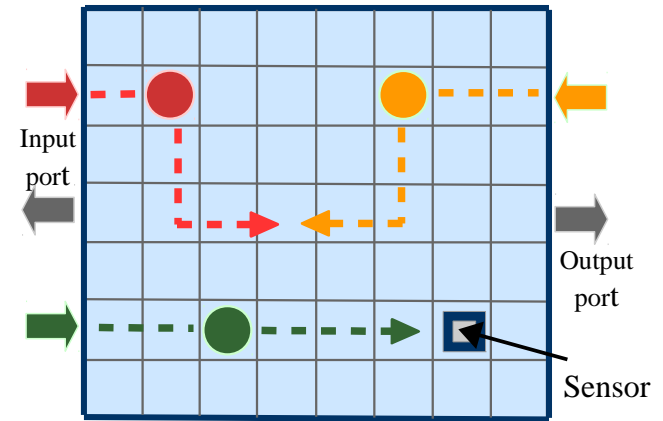
Fluidic Operations



DMB Architecture

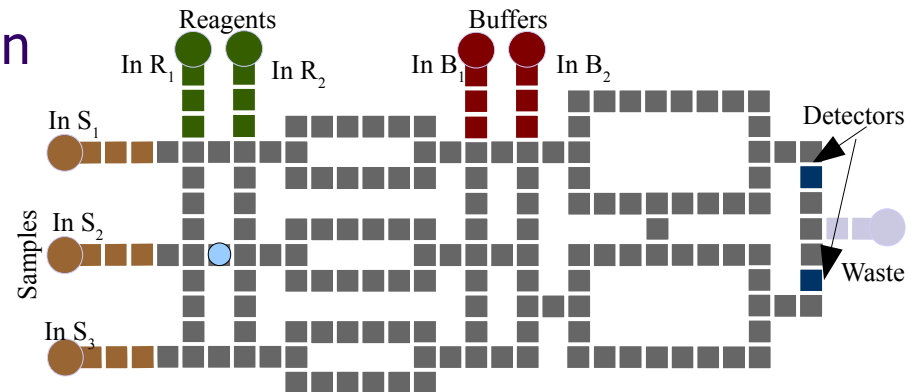
- General-Purpose Architecture

- Reconfigurable
- Versatile
- Fault-tolerant

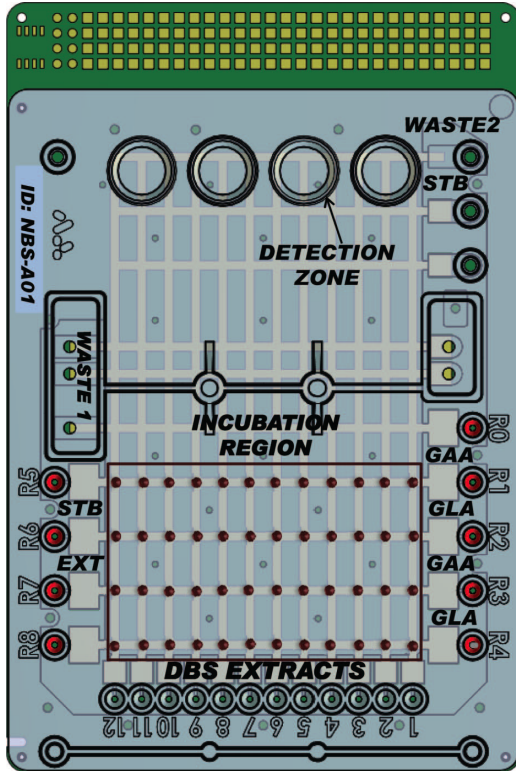


- Application-Specific Architecture

- Designed for one application
- Reduced costs
 - Production costs
 - Reagent costs

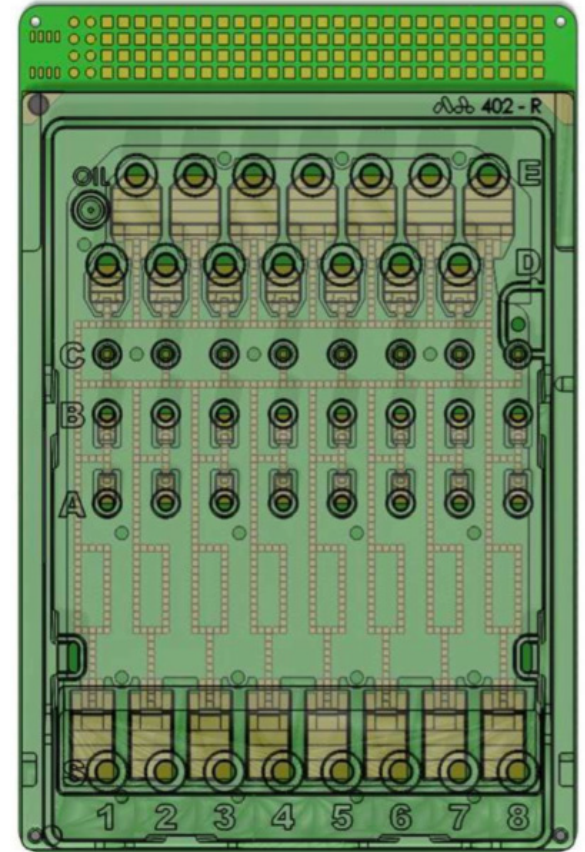


Application-Specific Biochips



Biochip for Newborn Screening

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



Biochip for Sample Preparation

<http://www.nugeninc.com/>

$$Cost_{\mathcal{A}} = \sum N_{M_i} \times Cost_{M_i}$$

where

- **A** is the architecture
- **N_{M_i}** is the number of components of type M
- **Cost_{M_i}** is the cost of the physical component M_i

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

Problem: Architecture Synthesis



- **Given**

- Biochemical application
- Deadline requirements
- Library of components (physical and virtual)
- The number k of permanent faults

- **Determine**

- An application-specific architecture \mathcal{A} , so that
 - the cost is minimized and
 - the application completes within deadline for any occurrence of the k faults

Optimization: Simulated Annealing

\mathcal{A}^0 - initial architecture

T^0 - initial temperature

T^L - temperature length

eps - cooling rate

temp = T^0 ;

$\mathcal{A} = \mathcal{A}^0$;

repeat

 while (temp < T^L) do

$\mathcal{A}^{new} = \text{moves}(\mathcal{A})$; *//generate new architecture*

 delta = Objective(\mathcal{A}) - Objective(\mathcal{A}^{best});

 if (delta < 0)

$\mathcal{A}^{best} = \mathcal{A}^{new}$;

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

$\mathcal{A}^{best} = \mathcal{A}^{new}$;

 endif

 endwhile

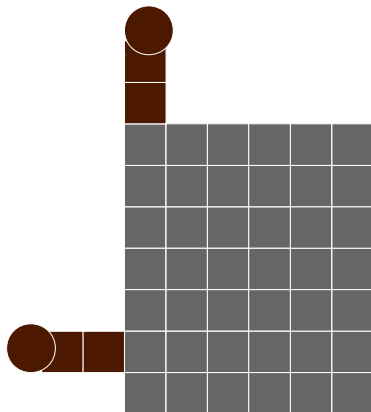
 temp = temp * eps;

until stop criterion is true

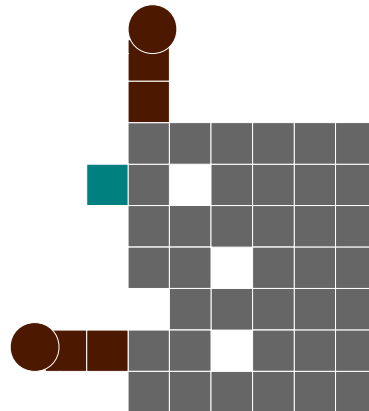
$$\text{Objective}(\mathcal{A}) = \text{Cost}_{\mathcal{A}} + W \times \max(0, \delta_G^k - D_G)$$

Optimization: SA moves

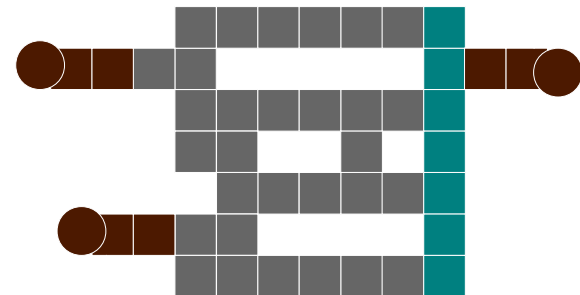
- Non-reconfigurable components (reservoirs, detectors)
 - Add/Remove
 - Change placement
- Reconfigurable elements (electrodes)
 - Add/Remove a single electrode
 - Add/Remove a row of electrodes on the side



Initial architecture

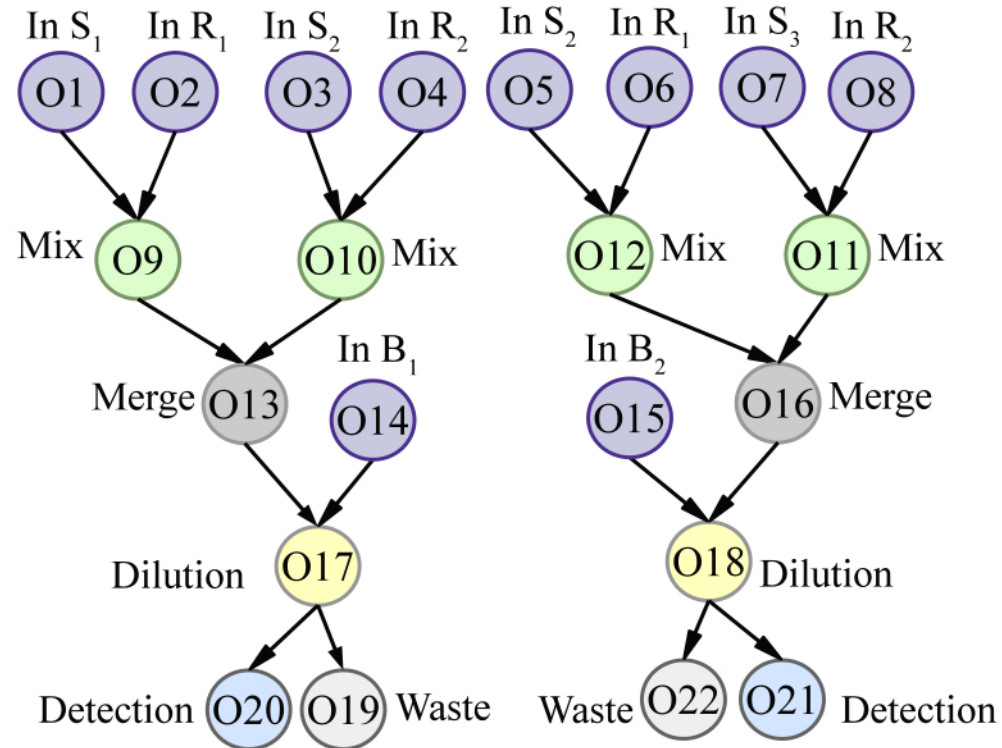


Add single electrode (green)
Remove single electrode (white)

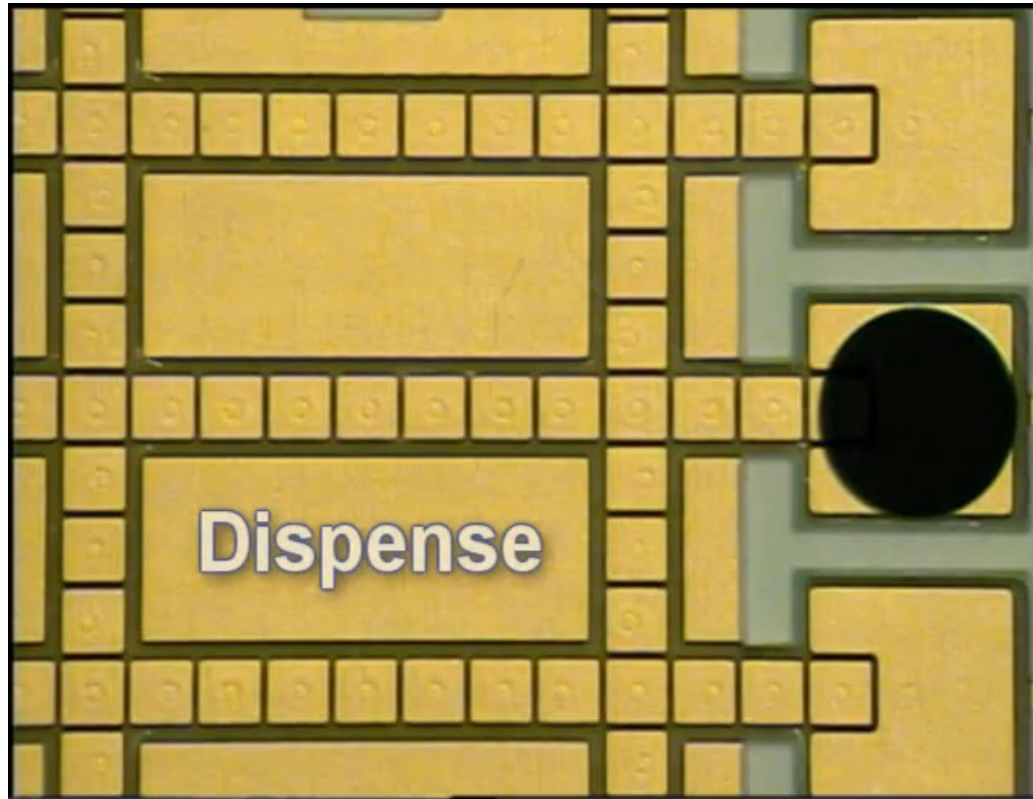


Add row of electrodes (green)
Change placement of reservoir (red)

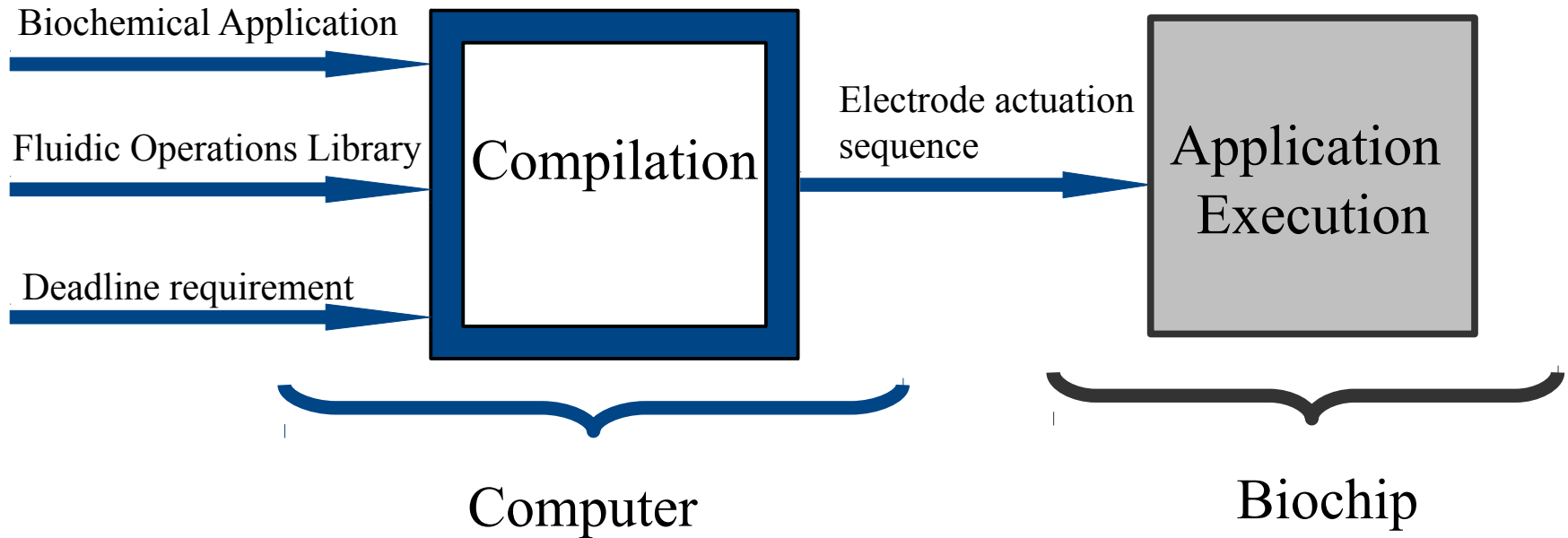
Biochemical Application Model



Electrode Actuation Sequence



Compilation Flow

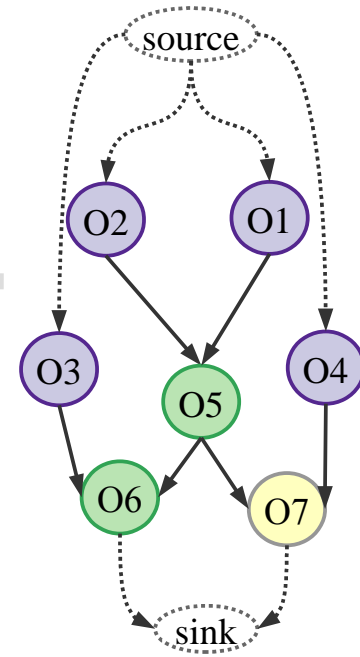


Compilation: Main steps

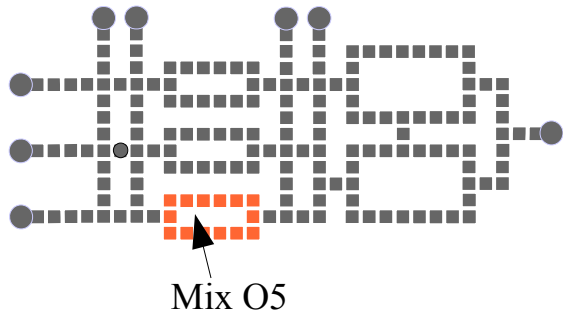
Allocation

Operation	Area	Time (s)
Mix	2x5	2
Mix	2x4	3
Mix	2x2	10
Detection	1x1	30

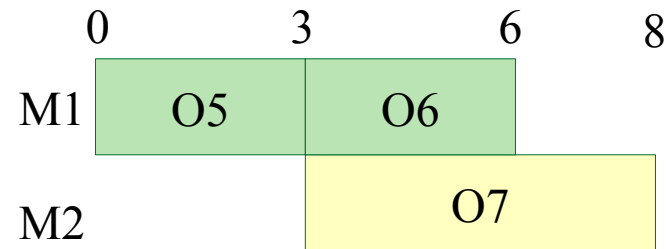
Binding



Placement

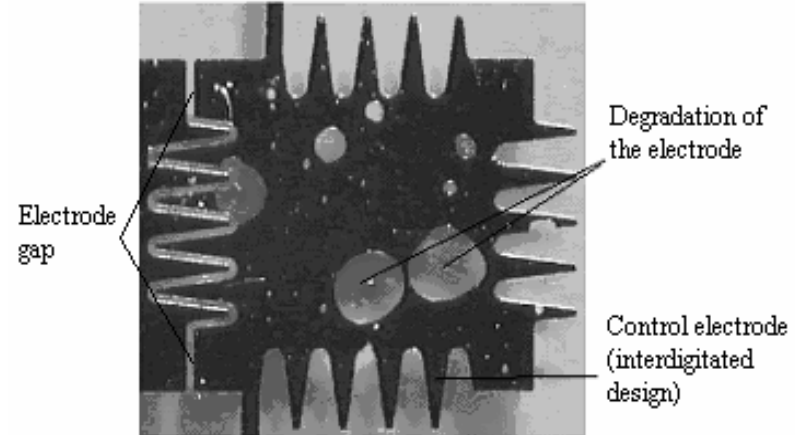


Scheduling



Compilation: difficulties

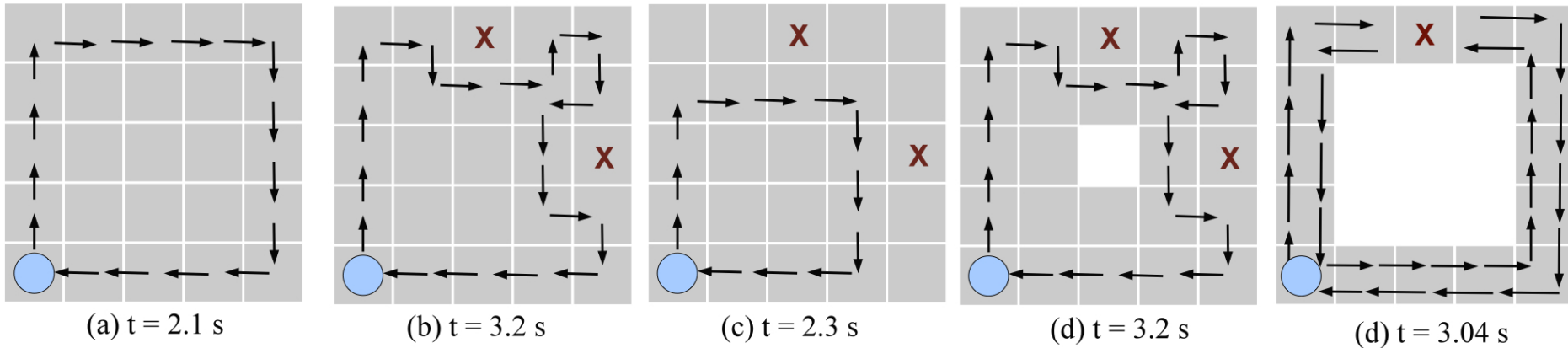
- **Problem:** permanent faults
- **Importance:**
 - Increase the yield of DMBs
 - Improve the batch control



Electrode degradation

- **Solution:**
 - Fault-tolerant overhead
 - Considers the impact of faults on the operation execution time

Fault-Tolerant Overhead



Evaluation of fault-tolerant overhead (faults are marked with X)

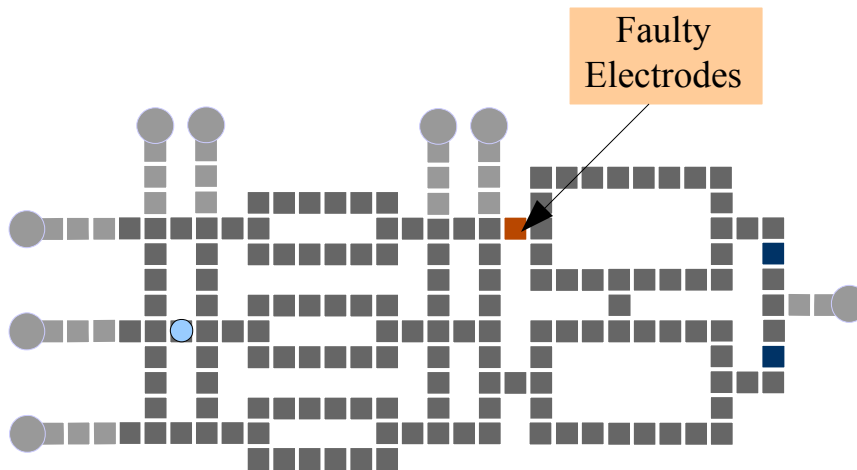
- Fault-tolerant overhead
 - Considers the impact of faults on the operation execution time
 - Routing-based operation execution, *Maftai 2012*

Compilation: difficulties

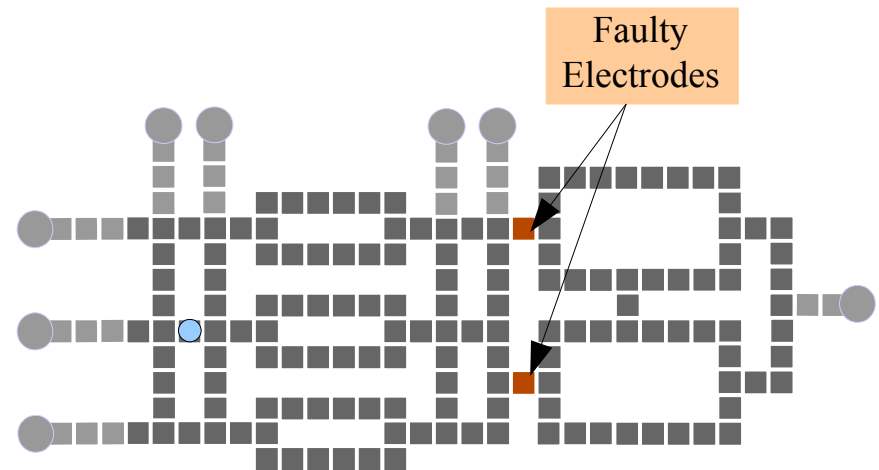


- **Problem:** fast compilation
- **Importance:**
 - It is part of an optimization loop
- **Solution:**
 - List-Scheduling based compilation
 - Routability test
 - Tests if, no matter where k faults are located, there is at least one route between any two electrodes

Routability test



Routable architecture for 1 fault

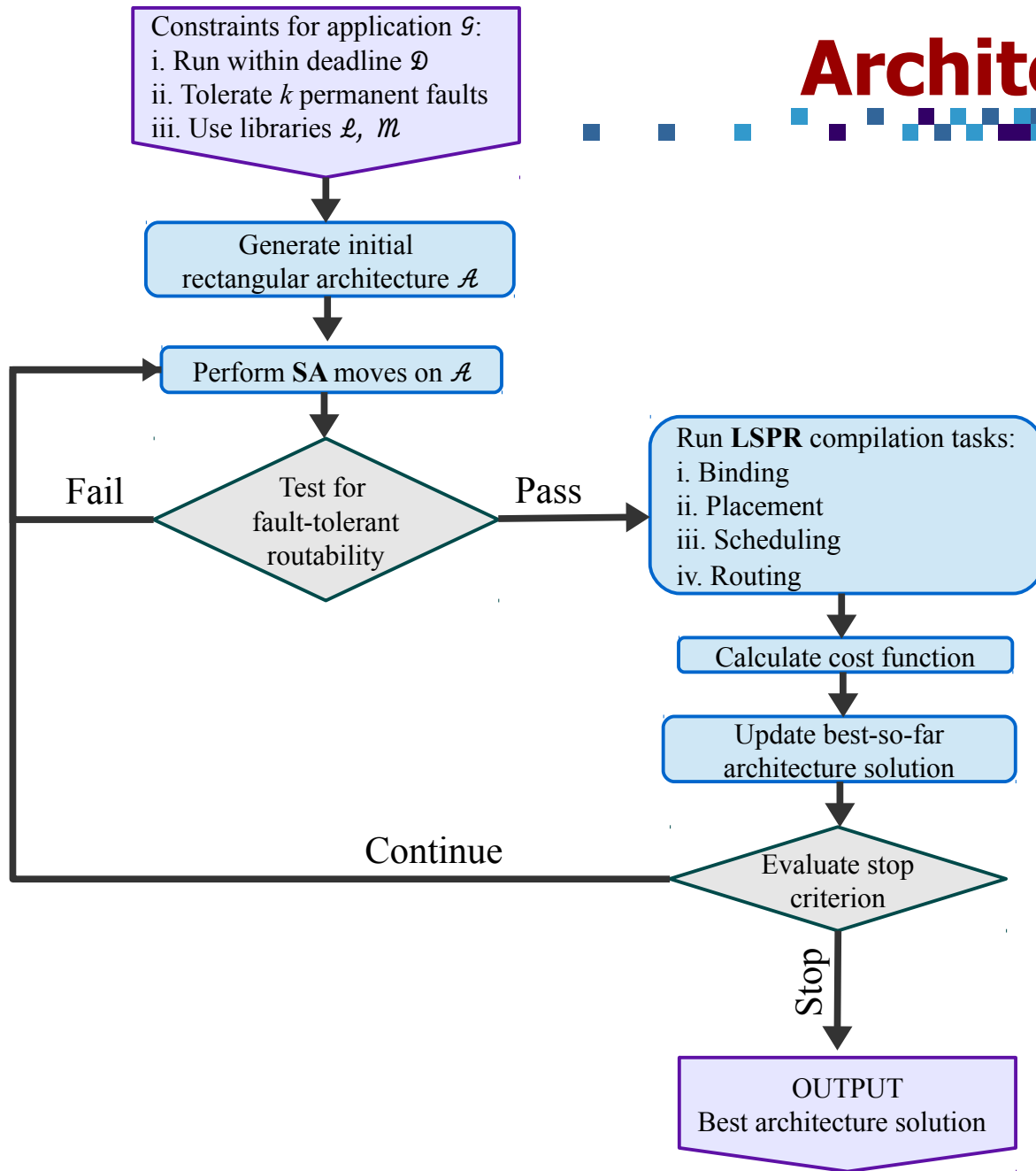


Non-Routable architecture for 2 faults

– Routability test

- Tests if, no matter where k faults are located, there is at least one route between any two electrodes
- Algorithm that tests k -vertex connectivity in a graph, *S. Even (1973)*

Architecture Synthesis



- Simulated Annealing
- Test k-vertex connectivity
- List-scheduling
- Fast-template placement
- Hadlock routing

- Biochemical applications:
 - The mixing stage of polymerase chain reaction (PCR)
 - In-vitro diagnosis on human physiological fluids (IVD)
 - The colorimetric protein assay (CPA)
- Deadlines:
 - PCR – 10 s; IVD – 15 s; CPA – 100 s
- Implementation:
 - Java
- **Evaluation:**
 - Pessimism of List-Scheduling based compilation
 - Overhead in execution time due to permanent faults ($k=0,1,2$)
 - Cost-effectiveness of the architectures resulted from our synthesis

Experiments: LS Compilation

App. (ops.)	Arch.	$\delta_G^0 (s)$	Exec. time	$\delta_G^{opt} (s)$	Exec. time	Deviation (%)
PCR (7)	9×9	11	25 ms	10	60 min	9
IVD (28)	9×10	77	91 ms	73	60 min	5.4
CPA (103)	11×12	219	498 ms	214	60 min	2.3

- Near-optimal value is obtained with Tabu-Search, *Maftei 2010*
- General-purpose architectures
- No faults
- **Average deviation from near-optimal is 5.5%**

Experiments: FT Overhead

App.	Cost	δ_G^0 (s)	δ_G^1 (s)	Deviation (%)	δ_G^2 (s)	Deviation (%)
PCR	98	8.42	8.81	4.6	9.43	11.9
IVD	85	12.62	13.11	3.8	14.81	17.3
CPA	129	153.9	169.3	10	190.11	23.5

- $k = 0$ faults (column 2)
- $k = 1$ faults (column 3)
- $k = 2$ faults (column 6)
- Applications are resulted from our synthesis
- **Average deviation from near-optimal is 11.8%**

Experiments: Architecture synthesis

App.	$k = 0$				$k = 1$				$k = 2$			
	Arch	Cost	C_{SA}	T_{SA}	Arch	Cost	C_{SA}	T_{SA}	Arch	Cost	C_{SA}	T_{SA}
PCR	7×10 (1,1,1)	79	60	14	7×10 (1,1,1)	79	65	38	9×11 (1,1,1)	108	98	50
IVD	7×10 (2,2,2)	88	62	16	7×10 (2,2,2)	88	70	58	10×8 (2,2,2)	98	85	45
CPA	7×8 (2,1,2)	71	59	10	7×8 (2,1,2)	71	66	20	11×12 (2,1,2)	147	127	30

- Applications are resulted from our synthesis (col. 4, 8, 12)
- General-purpose applications obtained by exhaustive search (col. 3, 7, 11)
- **Our synthesis produces cheaper architectures**

- SA-based architecture synthesis
 - List-Scheduling based compilation (fast)
- Reduced cost architectures
- Fault-tolerant architectures
- Increase the yield of DMBs

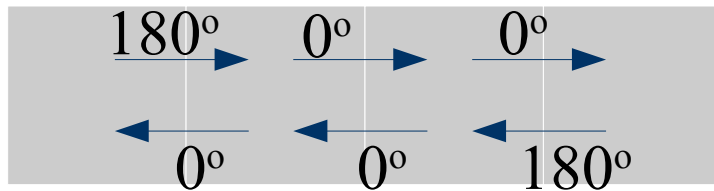
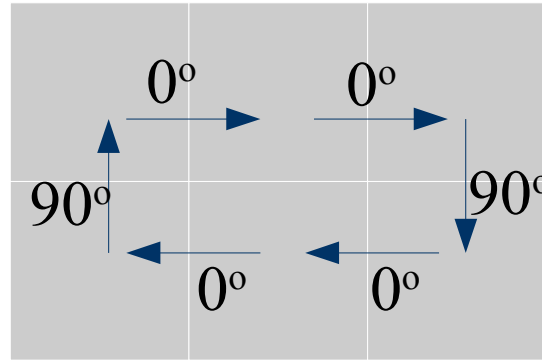
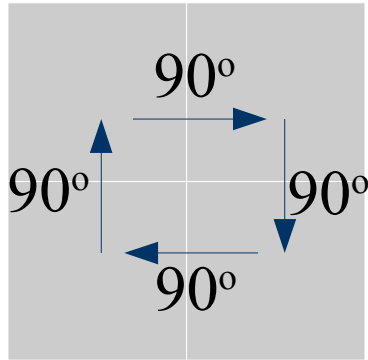
Backup slides



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 $t_i^{start} = \text{Schedule}(O_i, \mathcal{B}(O_i), \mathcal{C}, \mathcal{P})$
- 6 $t =$ earliest time when a scheduled operation terminates
- 7 $\text{UpdateReadyList}(Graph, t, List)$
- 8 **until** $List = \emptyset$
- 9 **return** \mathcal{S}

Routing-based Operation Execution



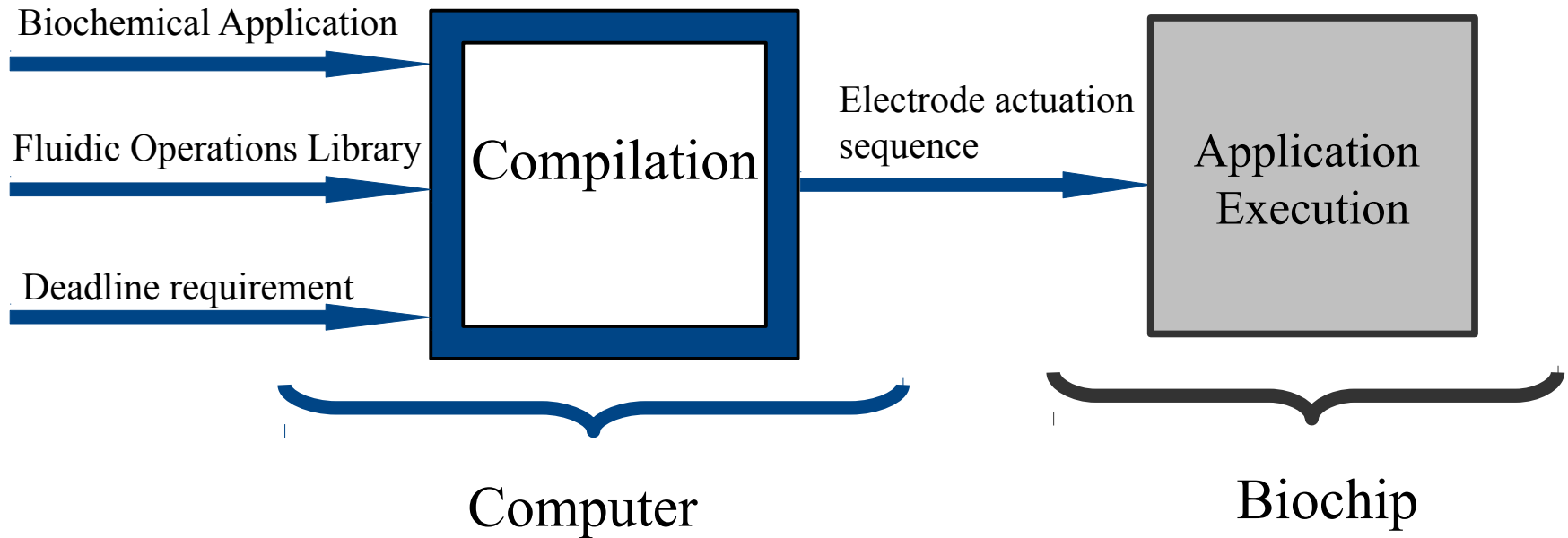
Operation	Area (cells)	Time (s)
Mix	2 x 2	10
Mix	2 x 3	6
Mix	1 x 4	5

$$p^{90} = 0.1\%$$

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

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

Compilation Flow

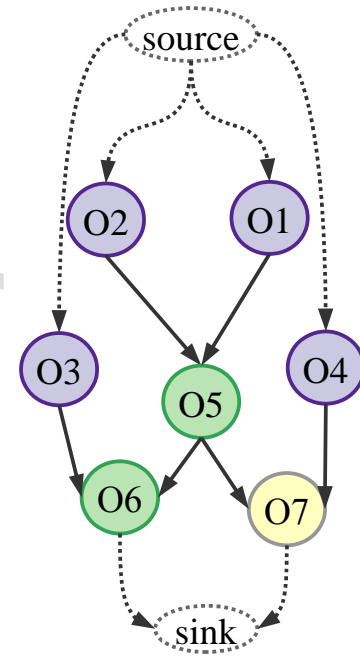


Compilation: Main steps

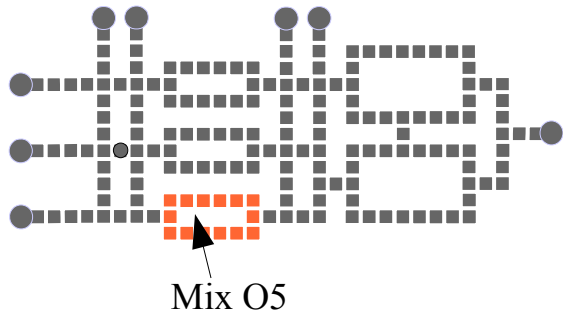
Allocation

Operation	Area	Time (s)
Mix	2x5	2
Mix	2x4	3
Mix	2x2	10
Detection	1x1	30

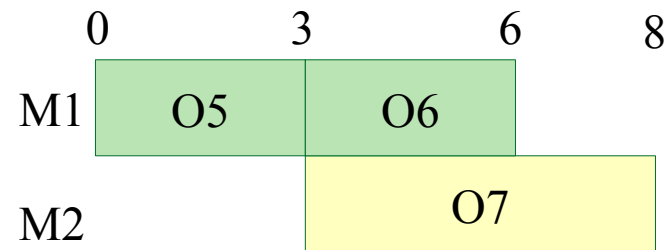
Binding



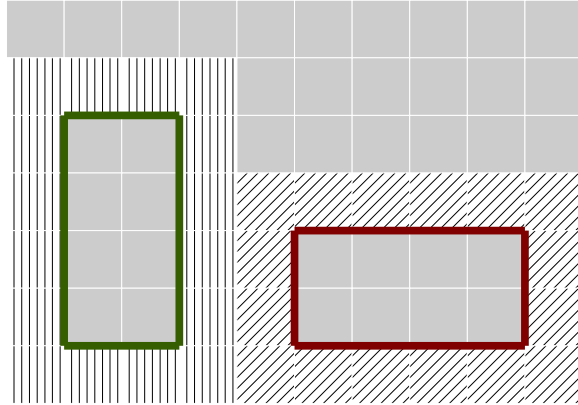
Placement



Scheduling

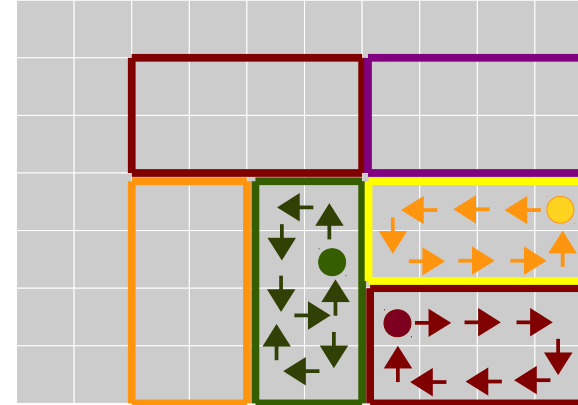


Droplet vs. Module Compilation



Module based

- module library
- black boxes
- protection borders



Droplet based

- routing based 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

