Design Automation for Flow-Based Biochips: Fault-Tolerant Design, Error Recovery and Experimental Validation

Paul Pop

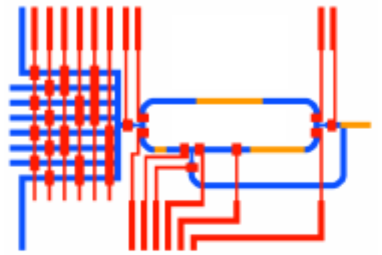
Technical University of Denmark

Outline

- Motivation
- System-level modeling
  - Biochip architecture
  - Application model
  - Fault model
- Techniques
  - Physical synthesis: flow and control layers
    - Compilation of high-level protocol languages
    - Design for fault-tolerance
  - Programming: application mapping
- Experimental validation

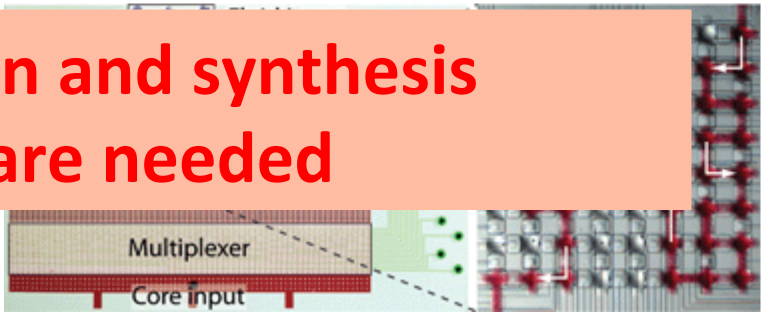
Physical design: current practice

- CAD tools in their infancy
  - Most groups use AutoCAD or Adobe Illustrator
  - Every line drawn by hand
  - No automation



New top-down design and synthesis methodologies are needed

postdoc time* [Fidalgo and Maerkl, Lab-on-a-chip, 2010]



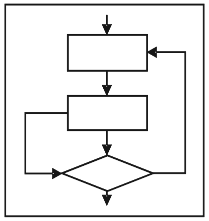
- Current practice
  - Tedious, time-consuming and error-prone
  - Required designer expertise
    - Understanding of application requirements
    - Knowledge and skills of chip design and fabrication

*Source: Philip Brisk, UCR

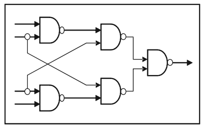
Physical design: VLSI vs mVLSI

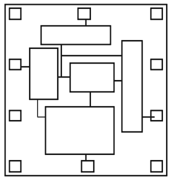
VLSI

$$\begin{tikzcd}
\text{System Specifications} \arrow[d] \\
\text{Architectural Design} \arrow[d] \\
\text{Functional Design} \arrow[d] \\
\text{Logic Design} \arrow[d] \\
\text{Circuit Design} \arrow[d] \\
\text{Physical Design} \arrow[d] \\
\text{Fabrication}
\end{tikzcd}$$



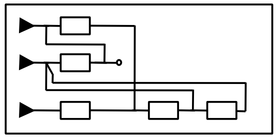
X = (AB + CD)
Y= (A(B+C))

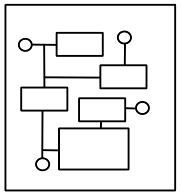


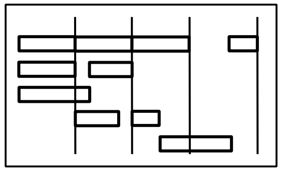


mVLSI

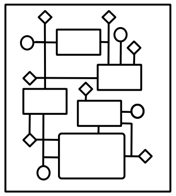
$$\begin{tikzcd}
\text{System Specifications} \arrow[d] \\
\text{Schematic Design} \arrow[d] \\
\text{Physical Design (Flow Layer)} \arrow[d] \\
\text{Application Mapping} \arrow[d] \\
\text{Control Synthesis} \arrow[d] \\
\text{Physical Design (Control Layer)} \arrow[d] \\
\text{Fabrication}
\end{tikzcd}$$





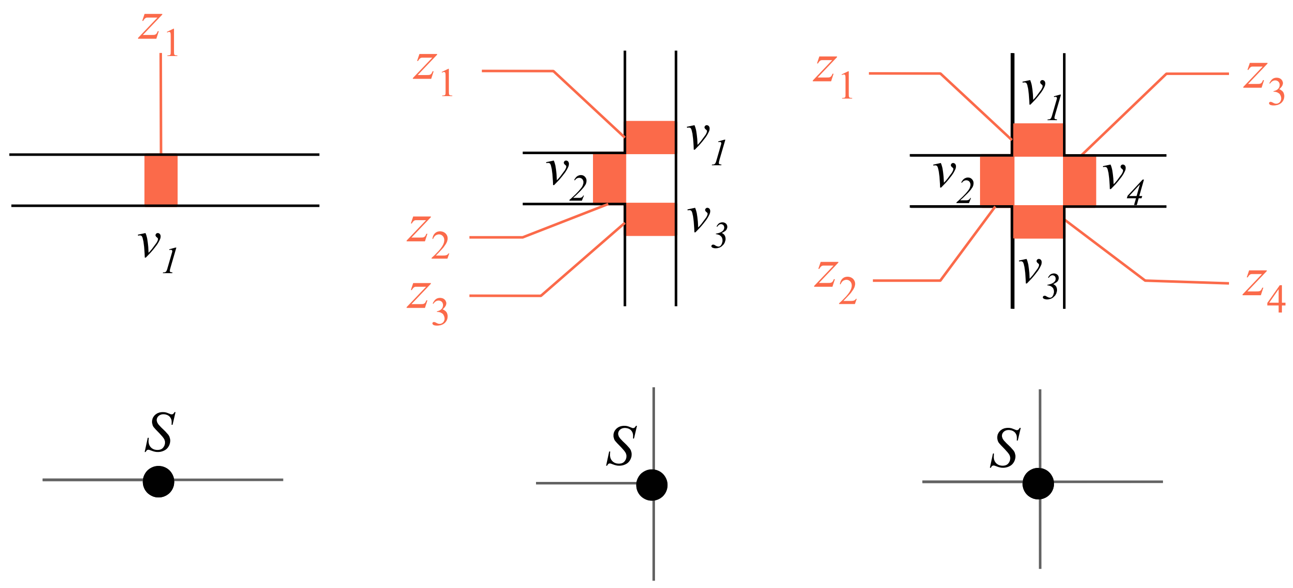


	t1	t2	t3	…
v1	0	1	0	…
v2	1	0	0	…
…	…	…	…	…



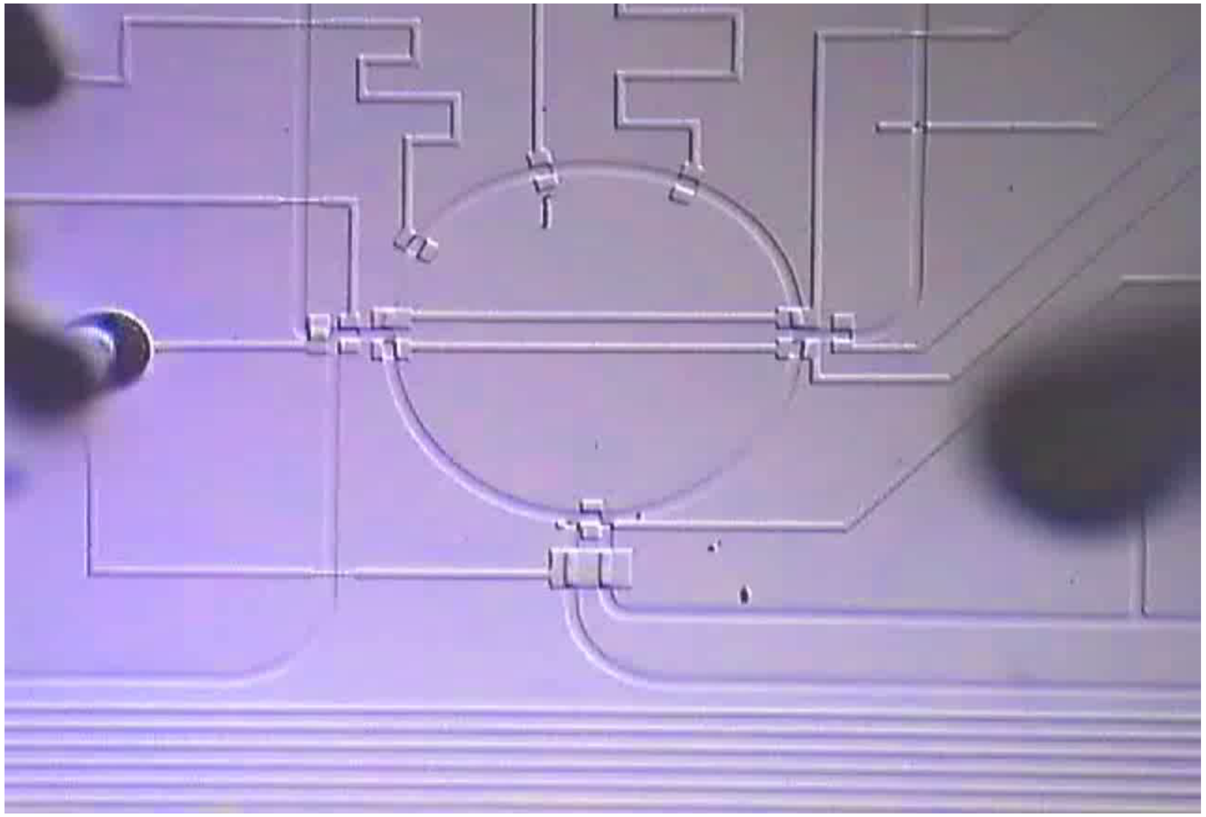
Components

Microfluidic switch



Components

Microfluidic mixer



http://groups.csail.mit.edu/cag/biostream

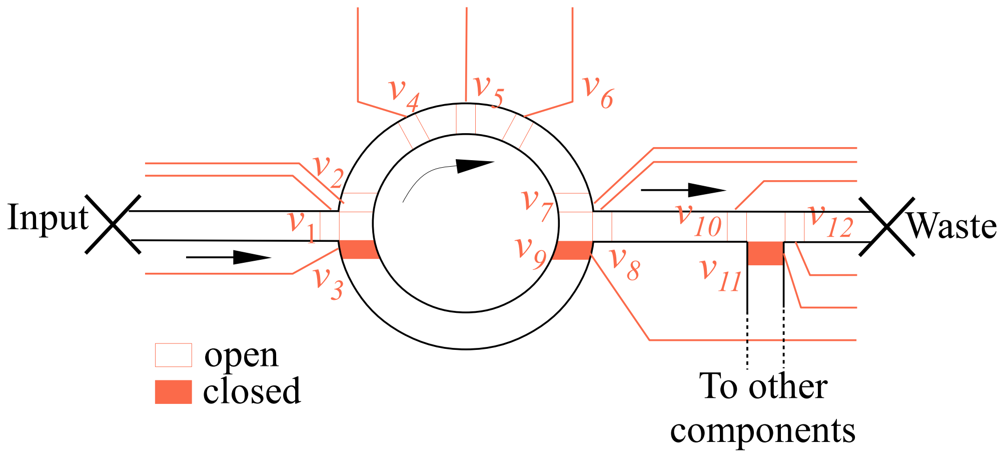
Component model

Microfluidic mixer

Flow layer model: Operational phases + Execution time

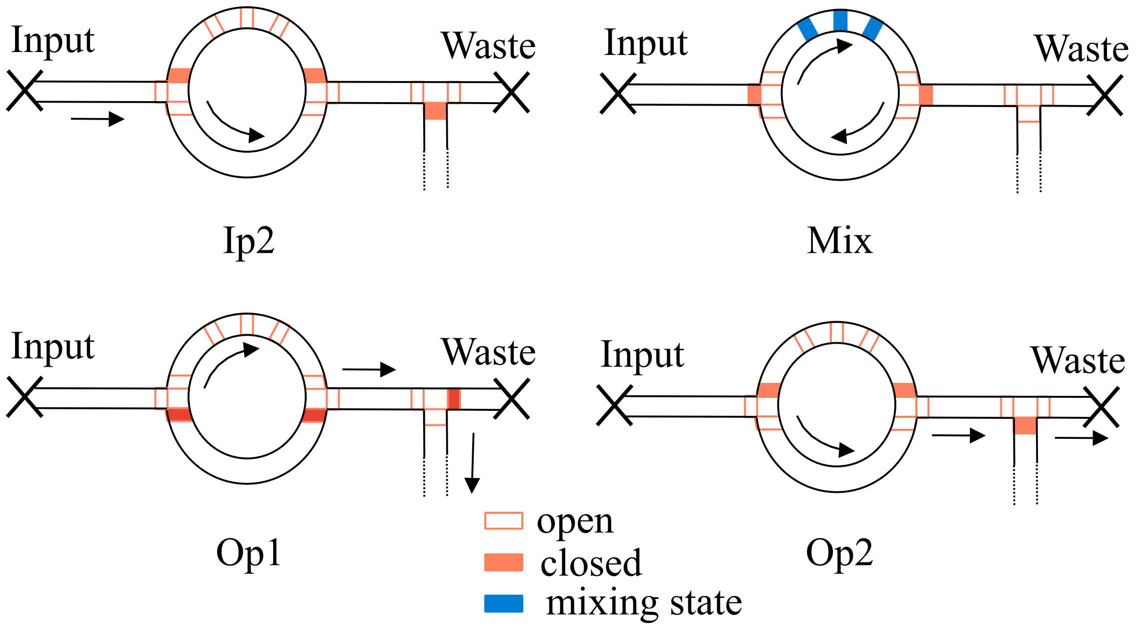
Five phases:

1. Ip1
2. Ip2
3. Mix (0.5 s)
4. Op1
5. Op2



Ip1

Component model

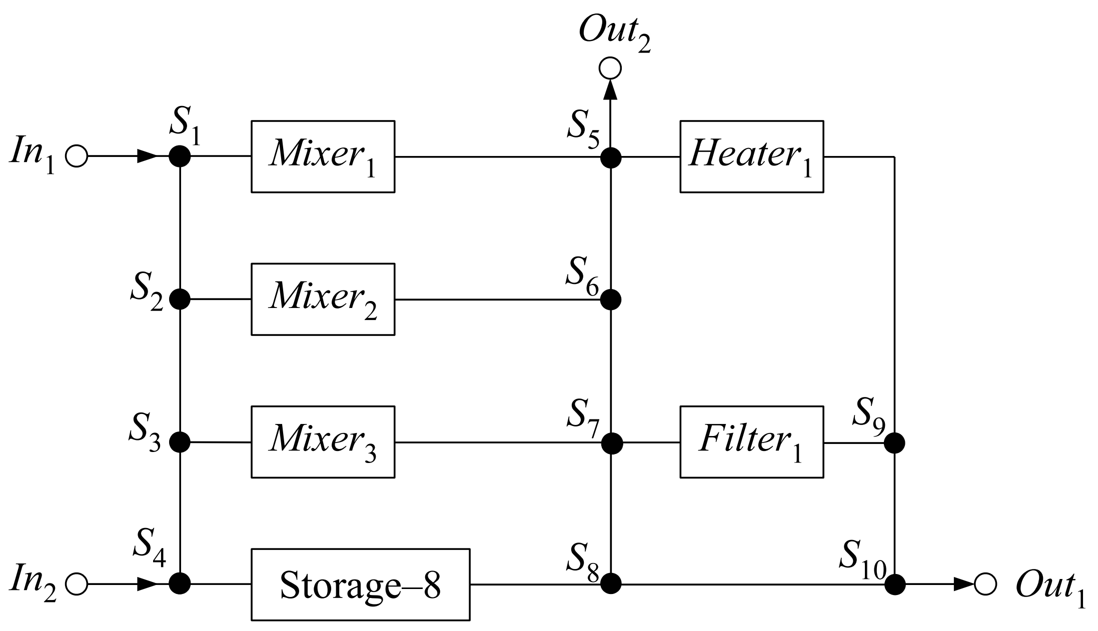


Control layer model

Phase	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
1. Ip1	0	0	1	0	0	0	0	0	1
2. Ip2	0	1	0	0	0	0	1	0	0
3. Mix	1	0	0	Mix	Mix	Mix	0	1	0
4. Op1	0	0	1	0	0	0	0	0	1
5. Op2	0	1	0	0	0	0	1	0	0

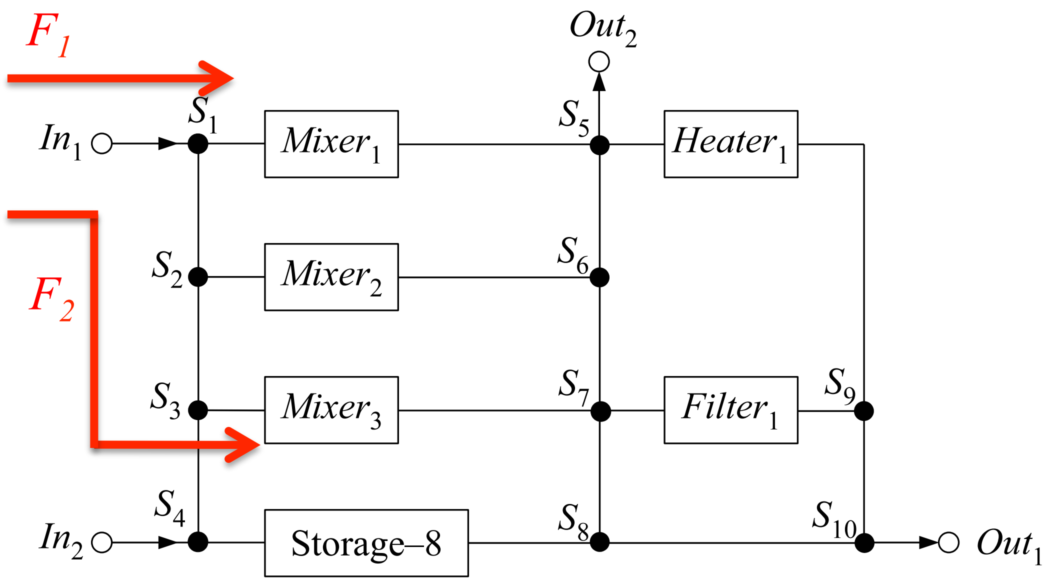
Biochip architecture model

Topology graph based model



Flow paths in the architecture

- Fluid transport latencies are comparable to operation execution times, so handling fluid transport (communication) is important
- **Routing constraints**: A flow path is reserved until completion of the operation, resulting in routing constraints



Problem formulation

- **Given**
  - A biochemical application (and a fault model)
  - Characterized component model library (including fault-tolerant components)
- **Synthesize**
  - A biochip architecture
  - Deciding on:
    - Component **allocation**
    - Schematic design and (and a fault-tolerant) ***netlist*** generation
    - Physical synthesis
      - **Placement** of components
      - **Routing** of microfluidic channels
  - **Such that**
    - the application completion time is minimized
    - Satisfying the fault-tolerance, dependency and resource constraints

Biochemical application modeling

- High-Level Languages (HLLs) for describing biological protocols
  - Biochemists describe protocol in natural language
  - Two languages proposed: Aqua and BioCoder/BioStream

Original protocol

Add 100 ul of 7X Lysis Buffer (Blue) and mix by inverting the tube 4-6 times.
Proceed to step 3 within 2 minutes.

BioStream code

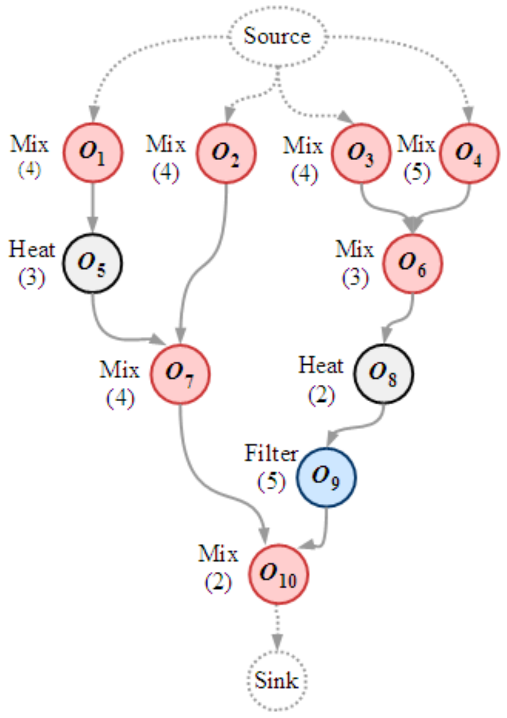
```
FluidSample f1 = measure_and_add(f0,
        lysis_buffer, 100*uL);
FluidSample f2 = mix(f1, INVERT, 4, 6);
time_constraint(f1, 2*MINUTES, next_step);
```

Aqua

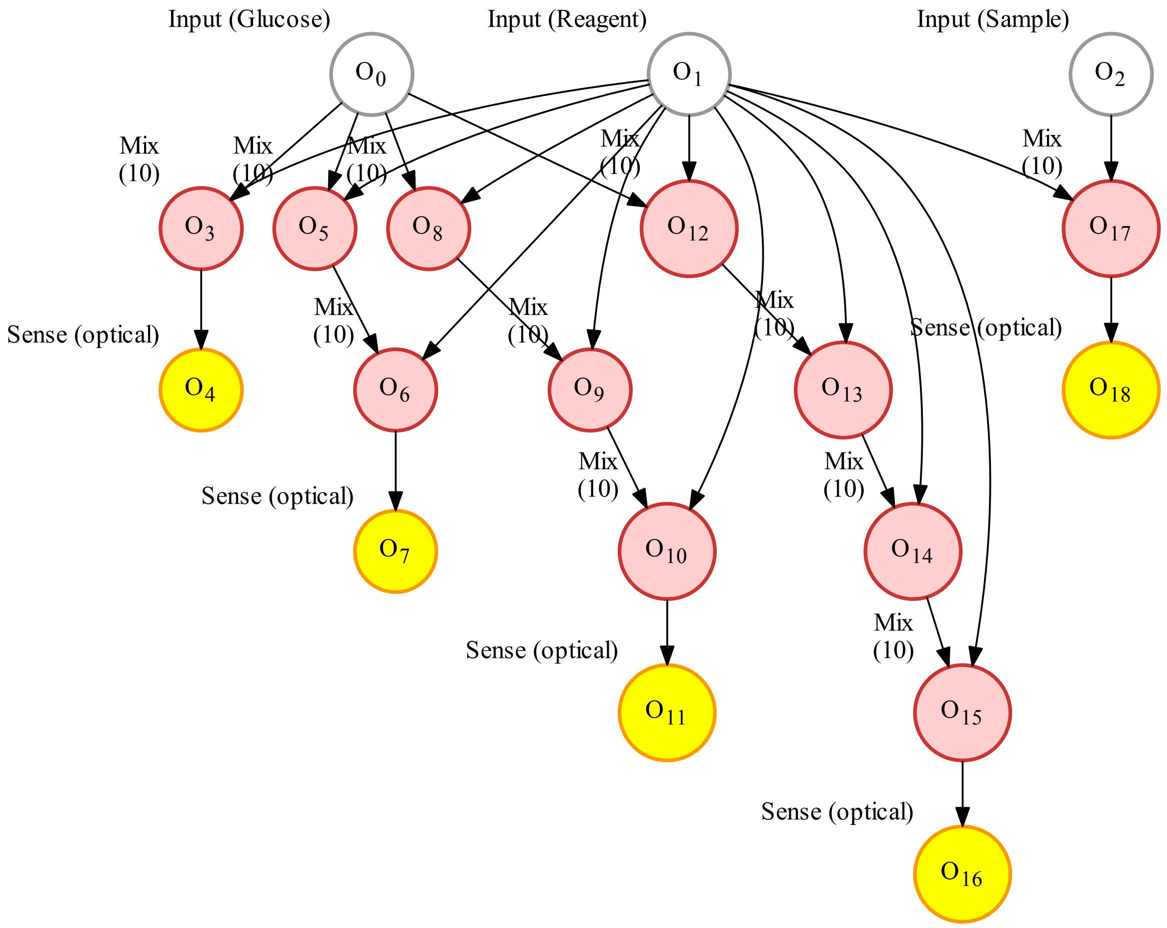
```
ASSAY Glucose START
fluid Glucose, Reagent, Sample;
fluid a, b, c, d, e;
VARResult[5];
input Glucose 50;
input Reagent;
input Sample 30;
conflict Sample FOLLOWS Glucose WASH water;
a=MIX Glucose AND Reagent IN RATIOS1 : 1 FOR 10;
SENSE OPTICAL it INTOResult[1];
b=MIX Glucose AND Reagent IN RATIOS1 : 2 FOR 10;
SENSEOPTICAL it INTOResult[2];
```

From HLLs to graph models

- We model a biochemical application as a graph

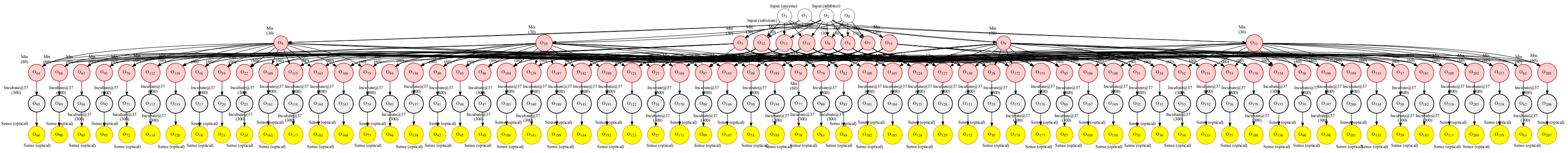


- Compiler: translates the protocol written in the HLL into the graph, doing mixing optimization

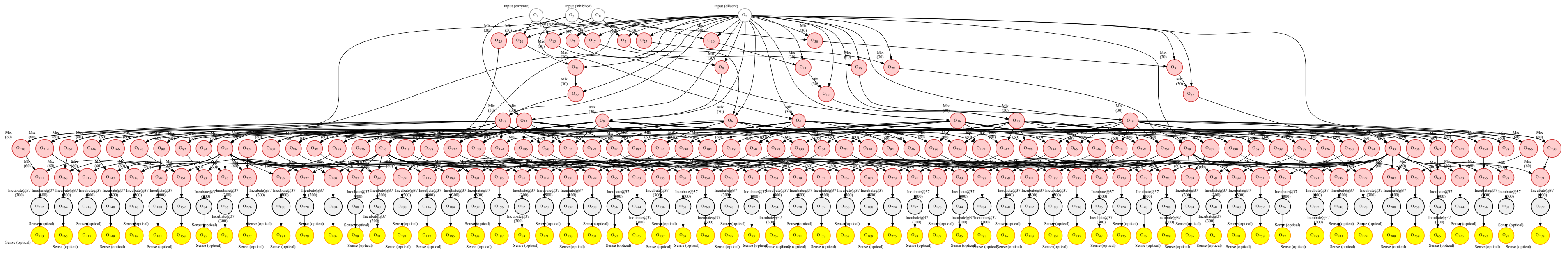


The graph model for an enzyme test

- Considering a **variable** mixer module

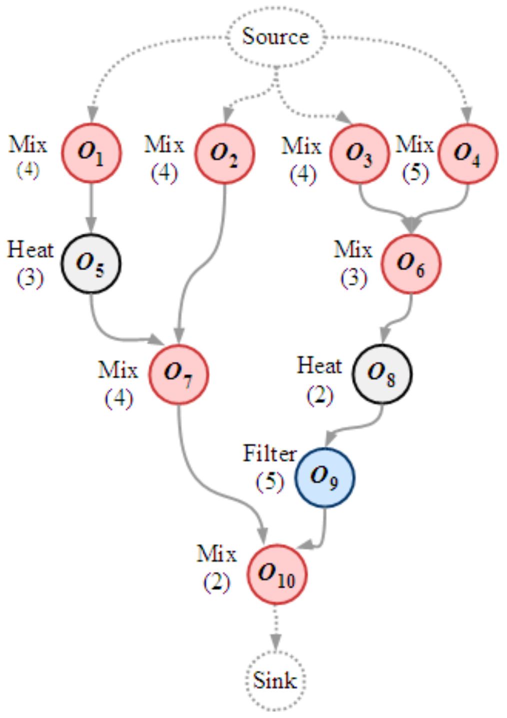


- Considering a **1:1 mixer module**; mixing is optimized

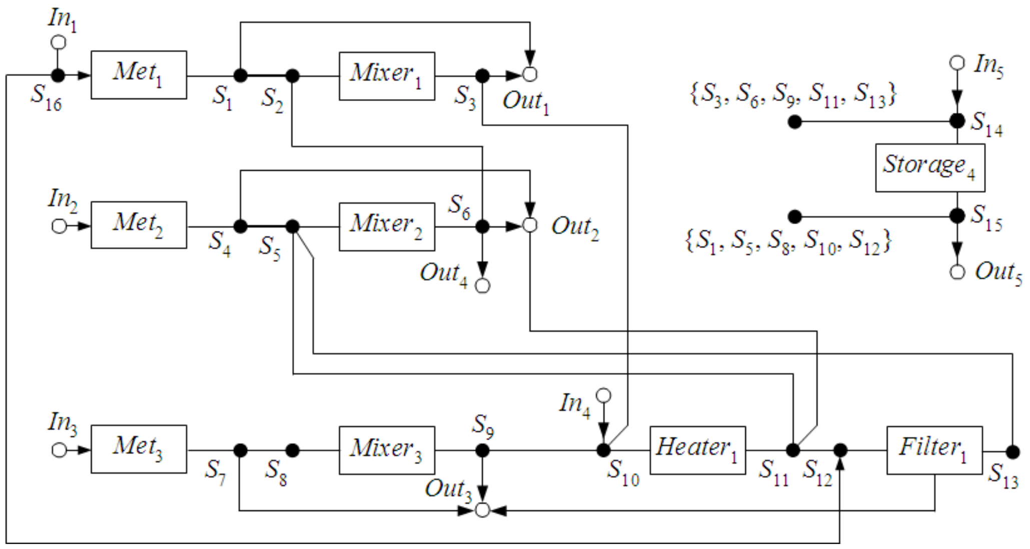


1) Allocation and schematic design

- How many components, and how to interconnect them?



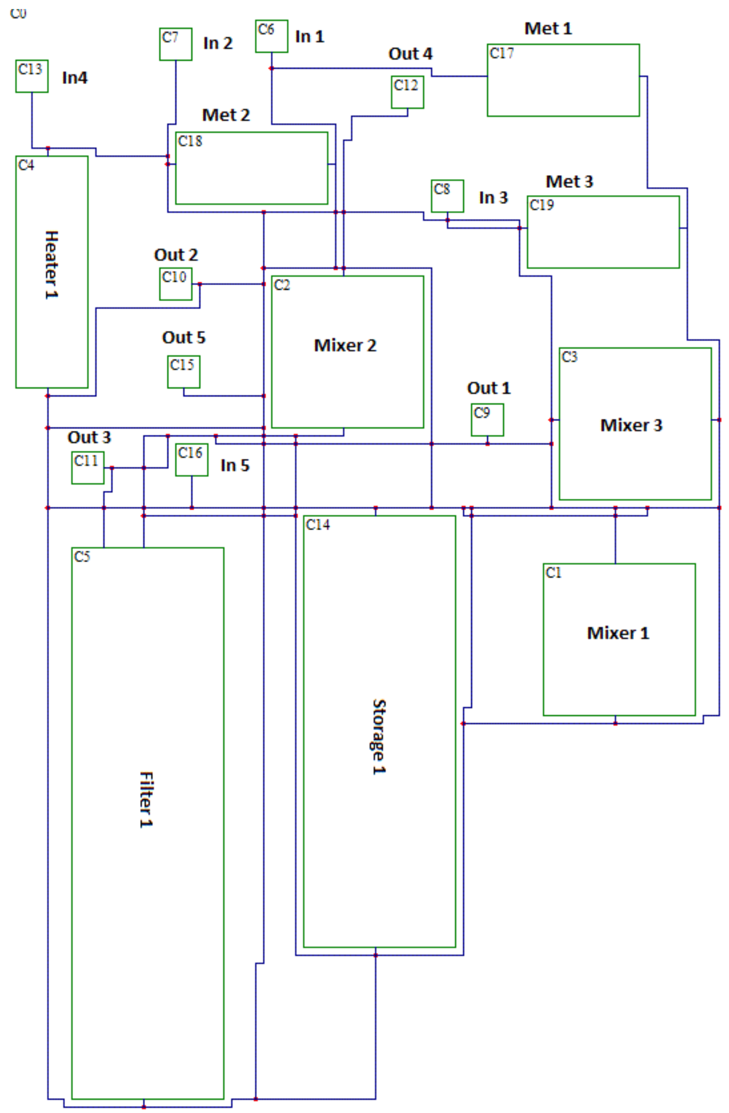
Mixer	3
Heater	2
Filter	1



- Input/ output ports
- Storage units
- Fluidic constraints

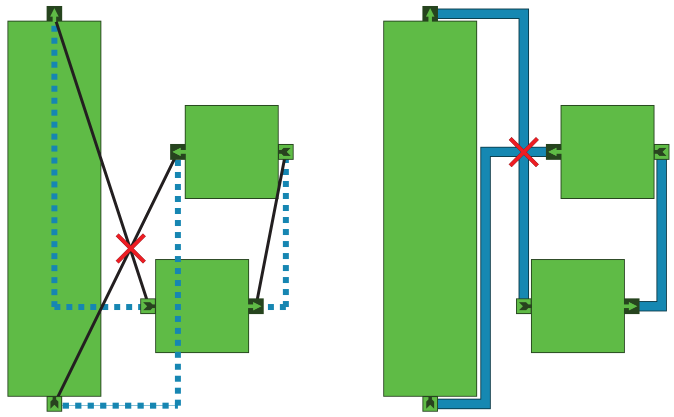
2) Physical synthesis – flow layer

- Placement
  - Where to place the components?
- Flow channel routing:
  - How to connect them?
- Extracting routing latencies



Placement

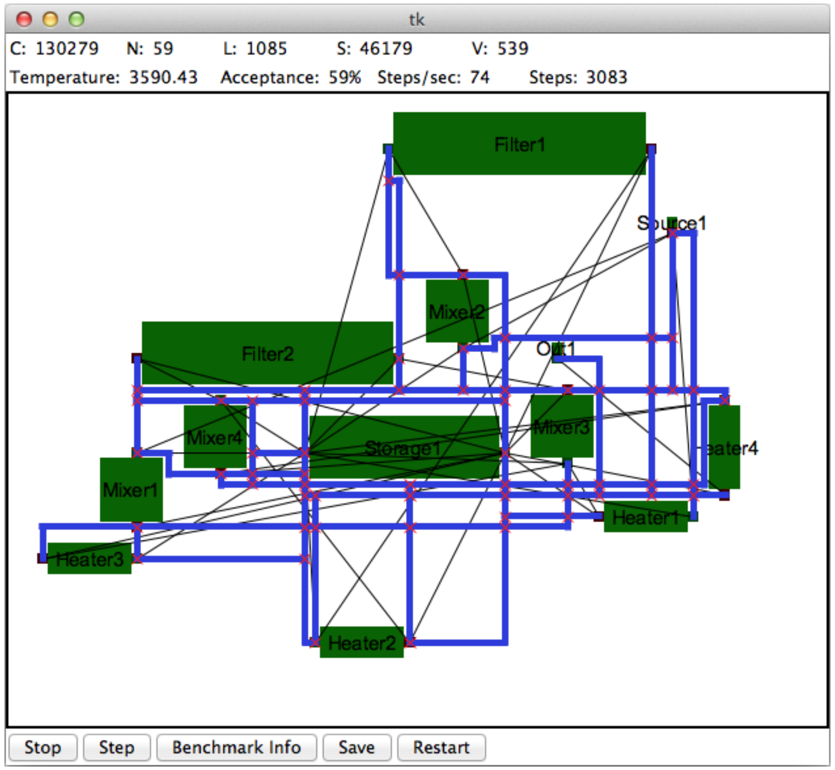
- "Simulated Annealing" (SA) implementations
  - Metaheuristic: uses "design transformations" and evaluates them
  - How to judge the quality of a placement?



(a) (b)

$$Cost_A(\mathcal{G}) = \alpha N_A + \beta L_A + \gamma S_A$$

actual routing



```
C: 130279    N: 59     L: 1085     S: 46179     V: 539
Temperature: 3590.43   Acceptance: 59%   Steps/sec: 74     Steps: 3083
```

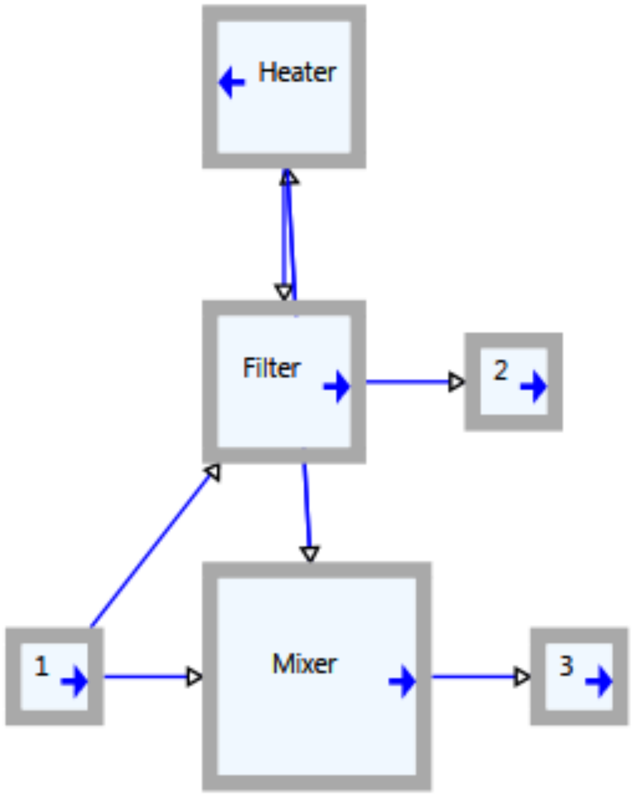
Cost functions:

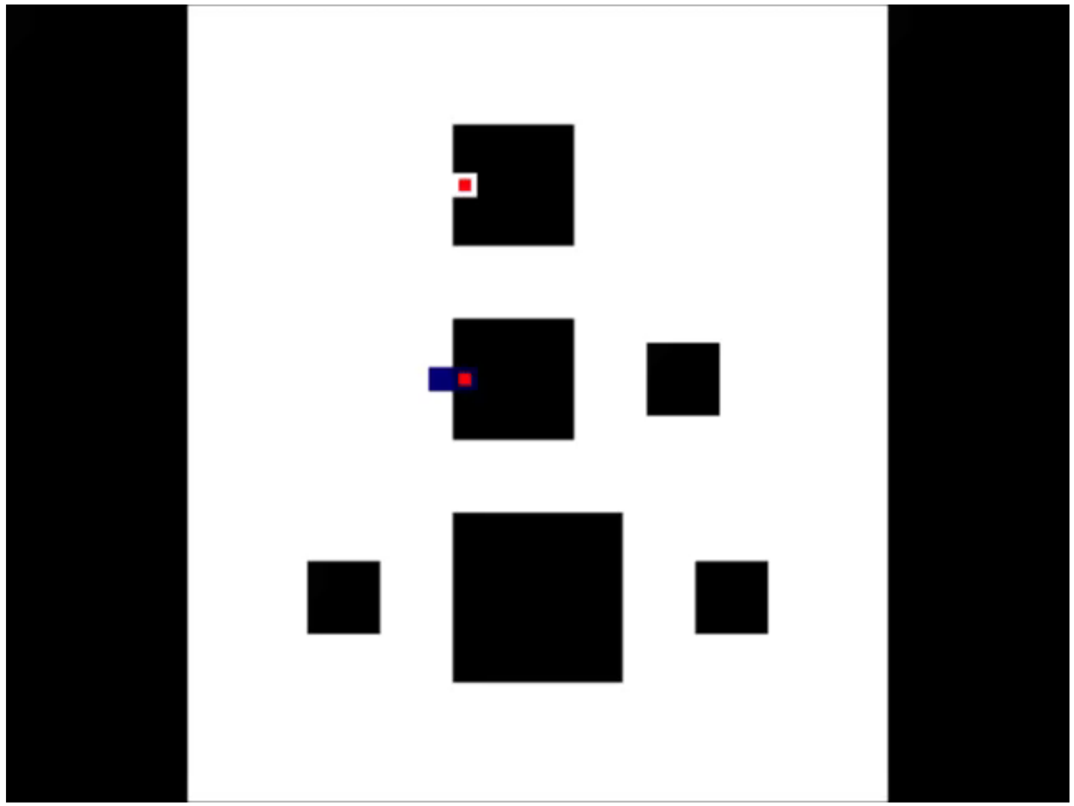
(a) no. of intersections + length + squared length

(b) actual routing lengths + no. of intersections

Flow-channel routing

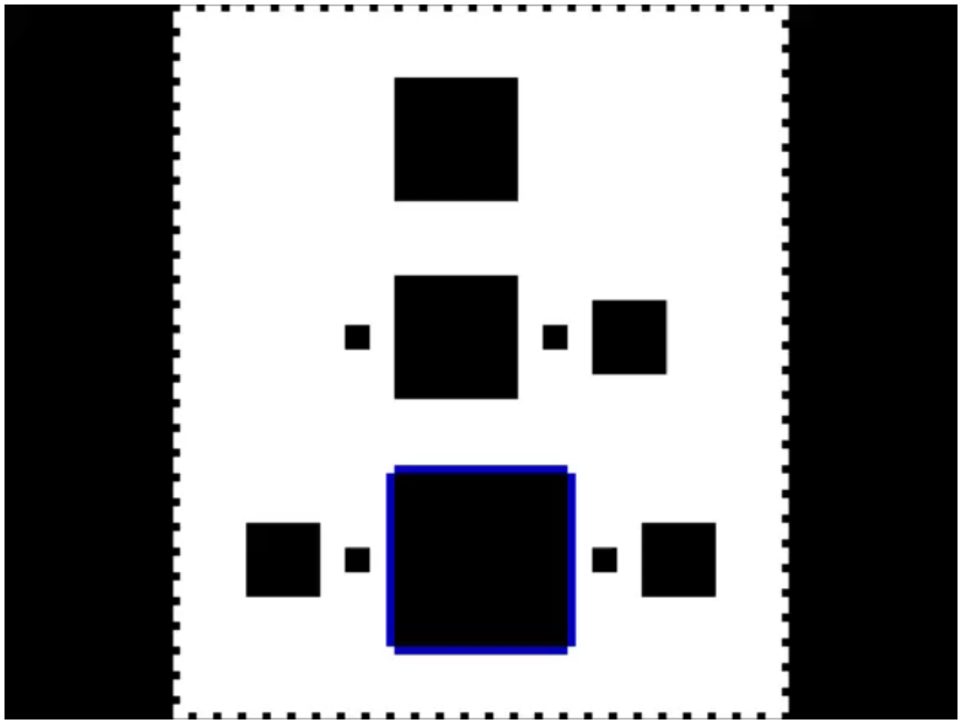
- Algorithms
  - Lee, Hadlock, Soukoup



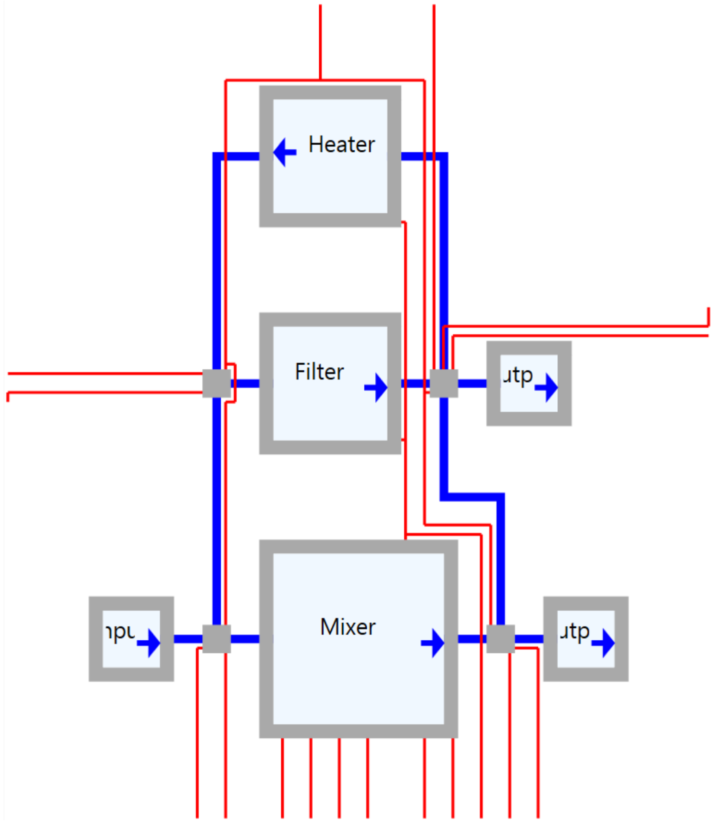


Control layer routing

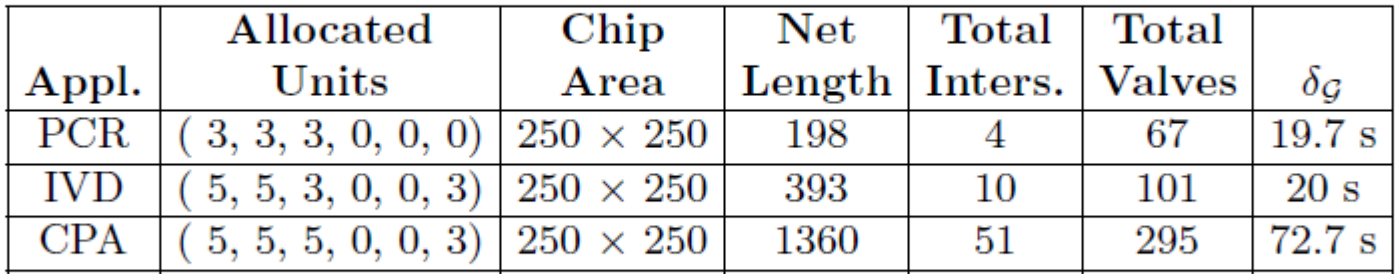
- Algorithms
  - Lee-Steiner: Route from component to nearest air inlet; rip-up and reroute
  - PathFinder



Lee-Steiner's algorithm



Results – real-life application



Appl.	Allocated Units	Chip Area	Net Length	Total Inters.	Total Valves	$\delta_{\mathcal{G}}$
PCR	(3, 3, 3, 0, 0, 0)	250 × 250	198	4	67	19.7 s
IVD	(5, 5, 3, 0, 0, 3)	250 × 250	393	10	101	20 s
CPA	(5, 5, 5, 0, 0, 3)	250 × 250	1360	51	295	72.7 s

PCR: Polymerase Chain Reaction mixing stage
IVD: In-Vitro Diagnostics
CPA: Colorimetric Protein Assay
Allocated units: (Input ports, output ports, Mixers, Heaters, Filters, Detectors)

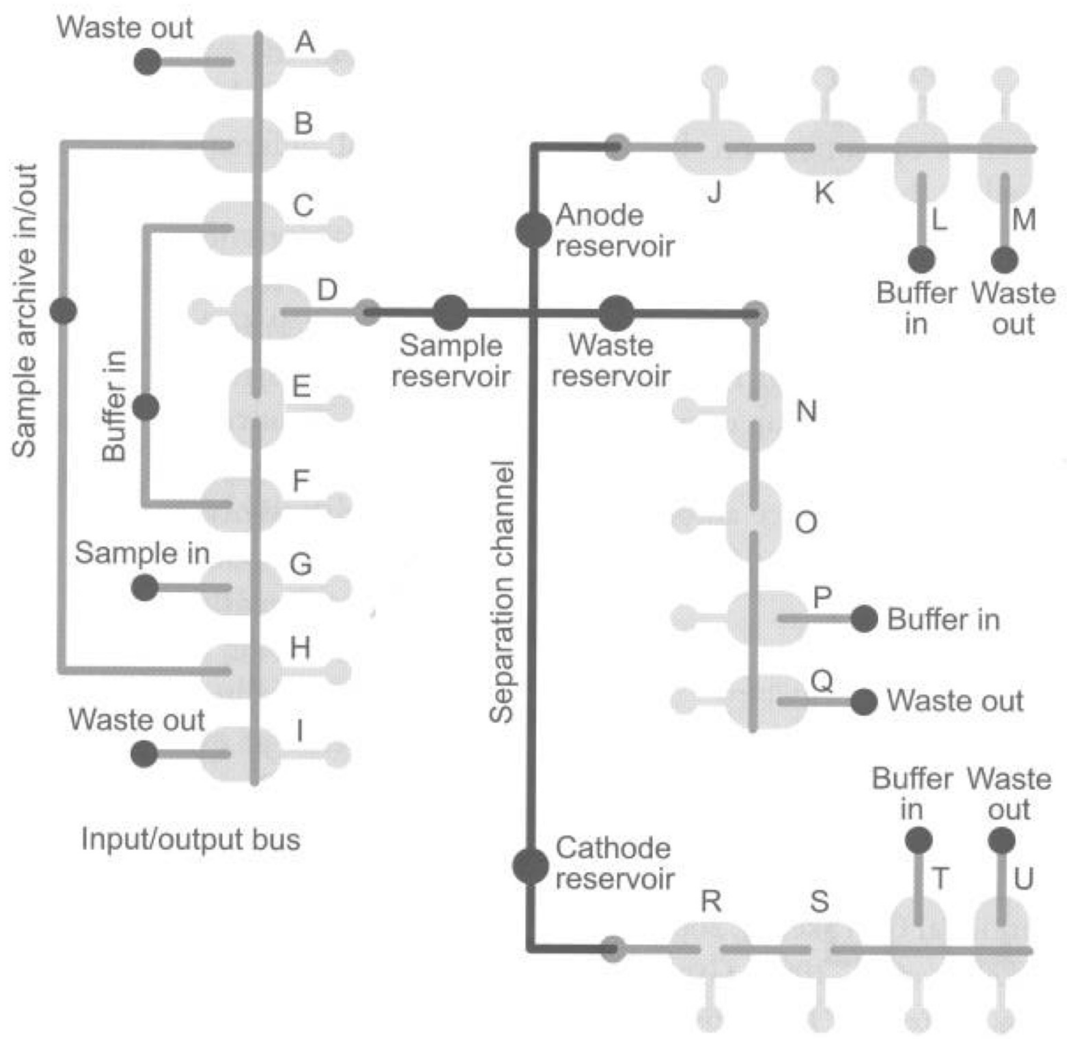
Design for fault-tolerance

- Design methodology steps
  1. Biochip design (as explained so far)
  2. Fabrication (the first talk)
  3. Testing (the previous talk)
  4. Defective chips are discarded

- Idea: introduce redundancy for fault tolerance
  - Increase the yield at the expense of chip area
    - A chip with a manufacturing fault can still be used
  - Redundancy could also be used to tolerate faults at “runtime”

- Part of the “Allocation and schematic design”
  - Output: fault-tolerant netlist, potentially using fault-tolerant components

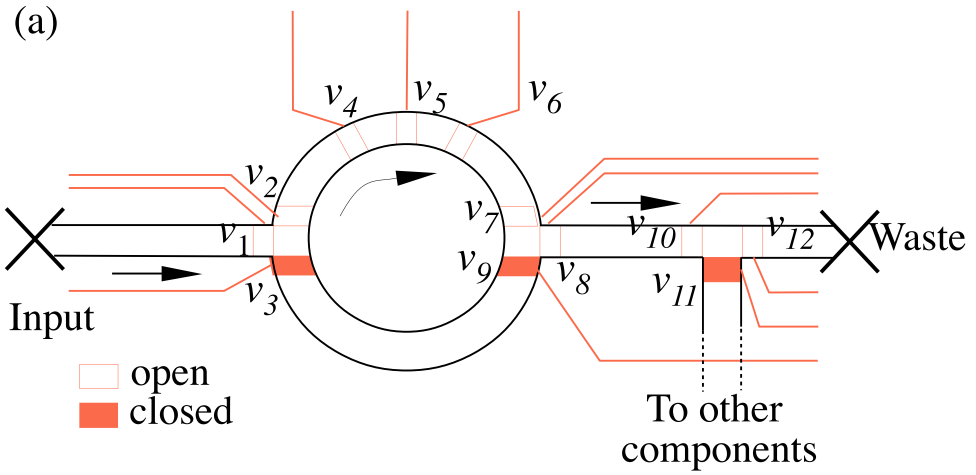
Redundancy is already used in practice

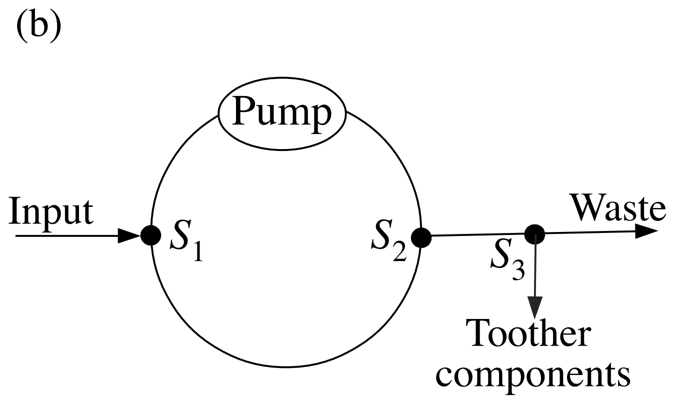


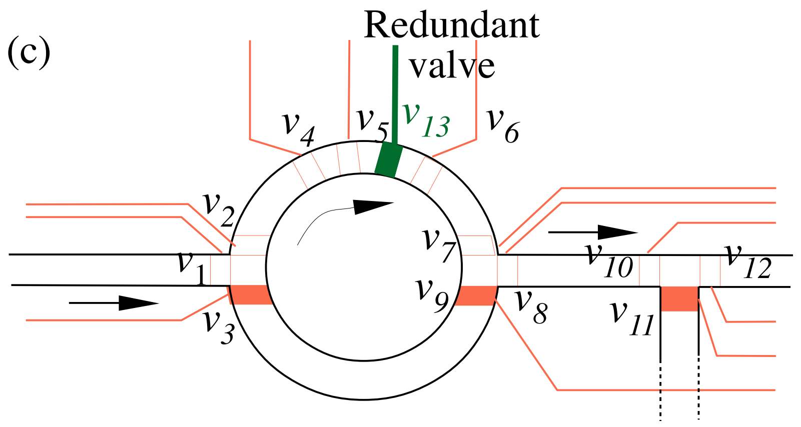
- Mars Organic Analyser chip developed for detecting biomolecules in Mars' soil
- Failure on Mars is extremely costly (no experiments will be possible if the chip fails)
- Redundancy: if pumping valve E fails, the sample can still be loaded in the CE reservoir via a redundant route consisting of valves G, H, B and D.

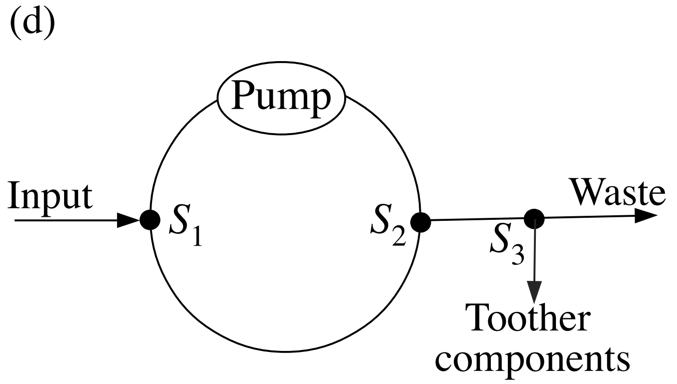
A. M. Skelley, J. R. Scherer, A. D. Aubrey, W. H. Grover, R. H. C. Ivester, P. Ehrenfre- und, F. J. Grunthaner, J. L. Bada, and R. A. Mathies. Development and evaluation of a microdevice for amino acid biomarker detection and analysis on Mars. Proceedings of the National Academy of Sciences of the United States of America, (4):1041–1046.

Fault-tolerant components

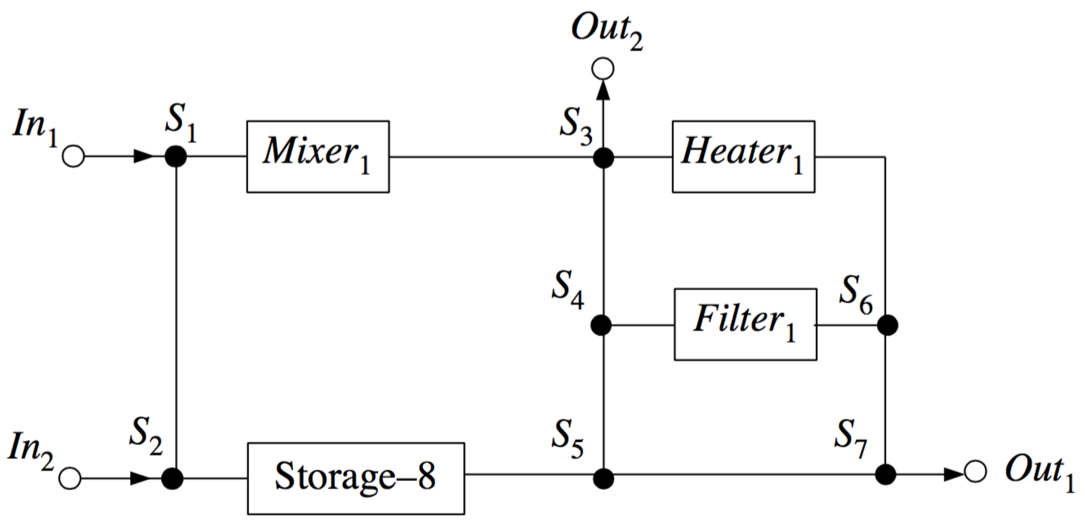




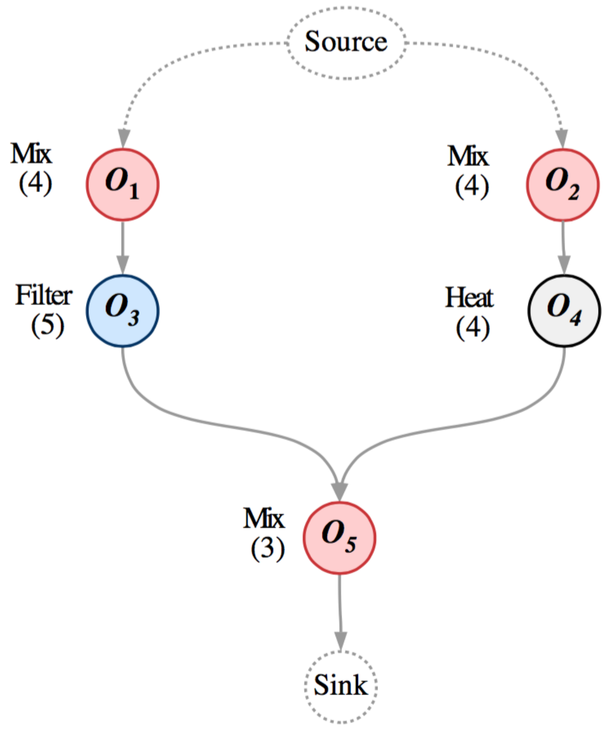




Design for fault-tolerance: motivation example



Architecture without fault-tolerance



Application

Fault model

- The designer gives the fault-model as an input: a set of possible faults; any combination may happen

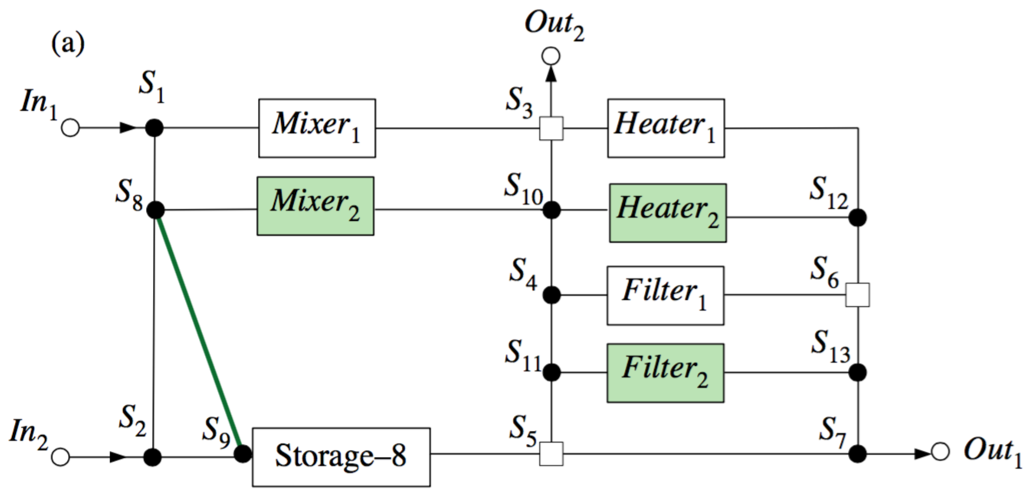
Table: The set of valve faults $\mathcal{VF}$

Name	Vertex ($N \in \mathcal{N}$)	Valve affected (w)	Type (t)
VF_1	$Mixer_1$	v_5	Open
VF_2	S_6	v_3	Open
VF_3	S_5	v_2	Open
VF_4	S_3	v_3	Open

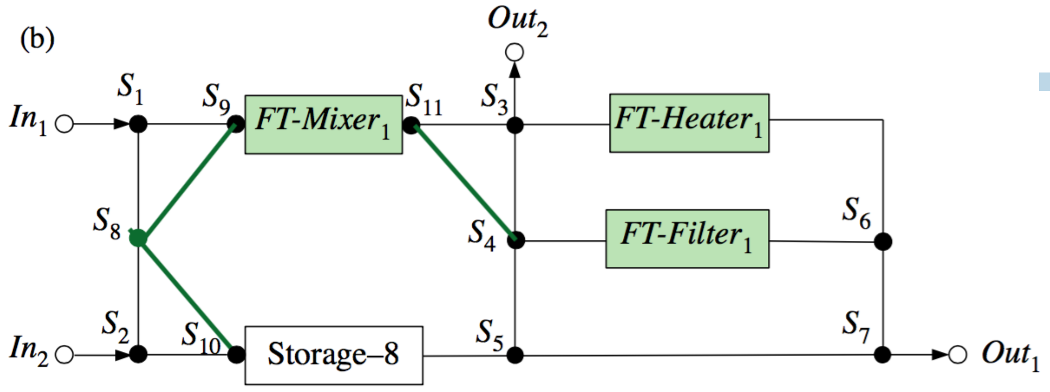
Table: The set of channel faults $\mathcal{CF}$

Name	Component ($M \in \mathcal{N}, \notin \mathcal{S}$) / Connection $D_{i,j} \in \mathcal{D}$	Type (t)
CF_1	$Heater_1$	Block
CF_2	$Filter_1$	Block
CF_3	$S_2 \rightarrow$ Storage-8	Block
CF_4	$S_1 \rightarrow Mixer_1$	Block

Straightforward vs. optimized redundancy

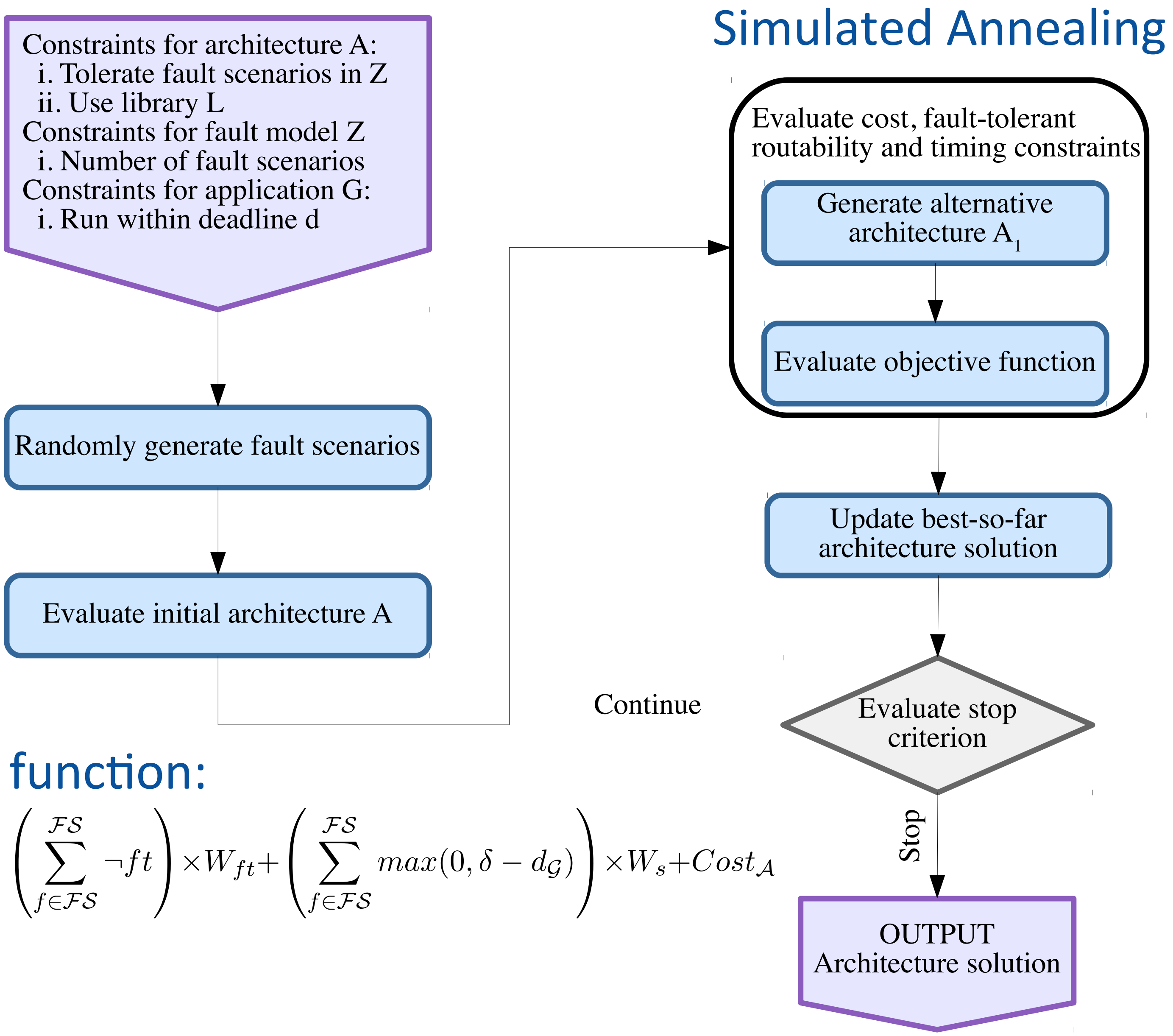


- Straightforward solution: redundancy not optimized; architecture cost: 129



- Optimized solution
  the introduction of redundancy is optimized; architecture cost: 96

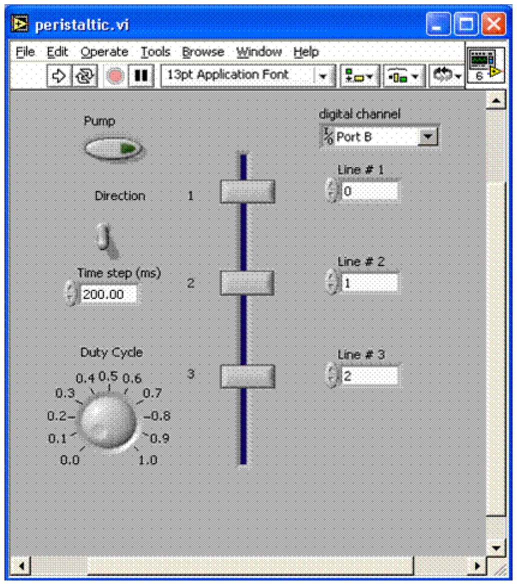
Synthesis of fault-tolerant netlists

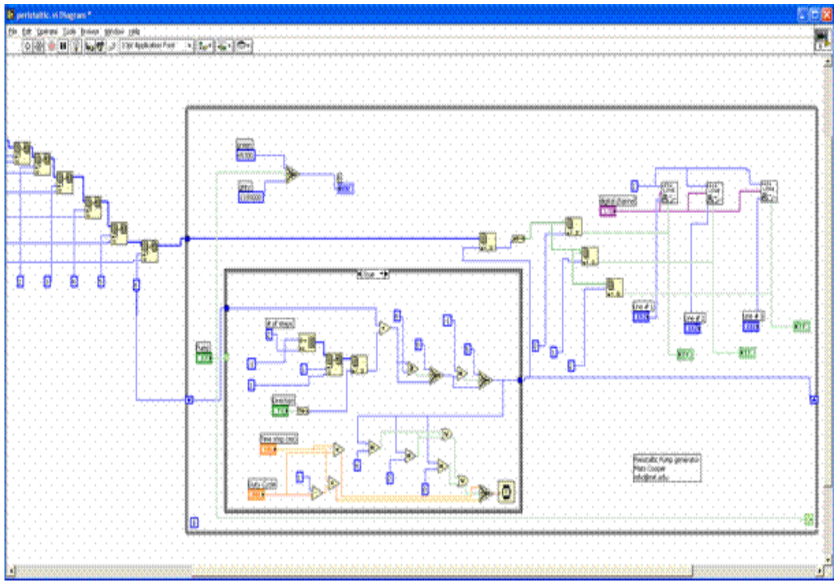

Simulated Annealing
Constraints for architecture A:
i. Tolerate fault scenarios in Z
ii. Use library L
Constraints for fault model Z
i. Number of fault scenarios
Constraints for application G:
i. Run within deadline d
Randomly generate fault scenarios
Evaluate initial architecture A
Evaluate cost, fault-tolerant routability and timing constraints
Generate alternative architecture A1
Evaluate objective function
Update best-so-far architecture solution
Evaluate stop criterion
Continue
Stop
OUTPUT
Architecture solution

Objective function:

$$Objective(\mathcal{A}) = \left(\sum_{f \in \mathcal{FS}}^{\mathcal{FS}} \neg ft\right) \times W_{ft} + \left(\sum_{f \in \mathcal{FS}}^{\mathcal{FS}} max(0, \delta - d_{\mathcal{G}})\right) \times W_s + Cost_{\mathcal{A}}$$

Application mapping: current practice



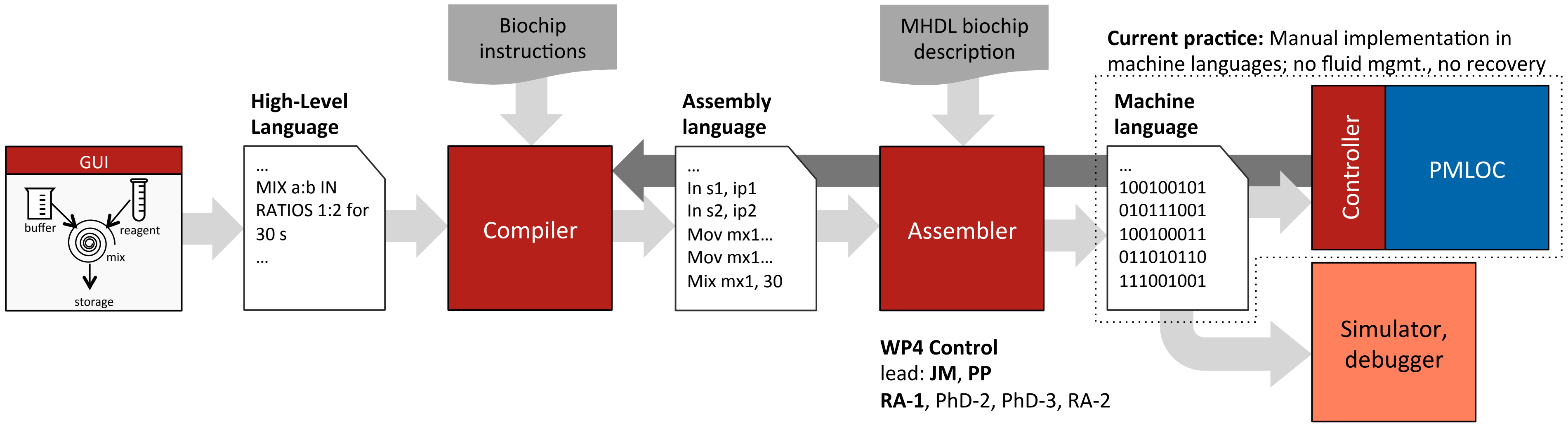


- Manually map experiments to the valves of the device
  - Using Labview or custom C interface
  - Given a new device, start over and do mapping again
  - With complexity increasing, the method becomes inadequate
- Similar to having gate-level exposed to the user in VLSI

Slide source: Bill Thies, Microsoft Research, India

Programming biochips: vision

- Programming tools: compiler, assembler, debugger, simulator

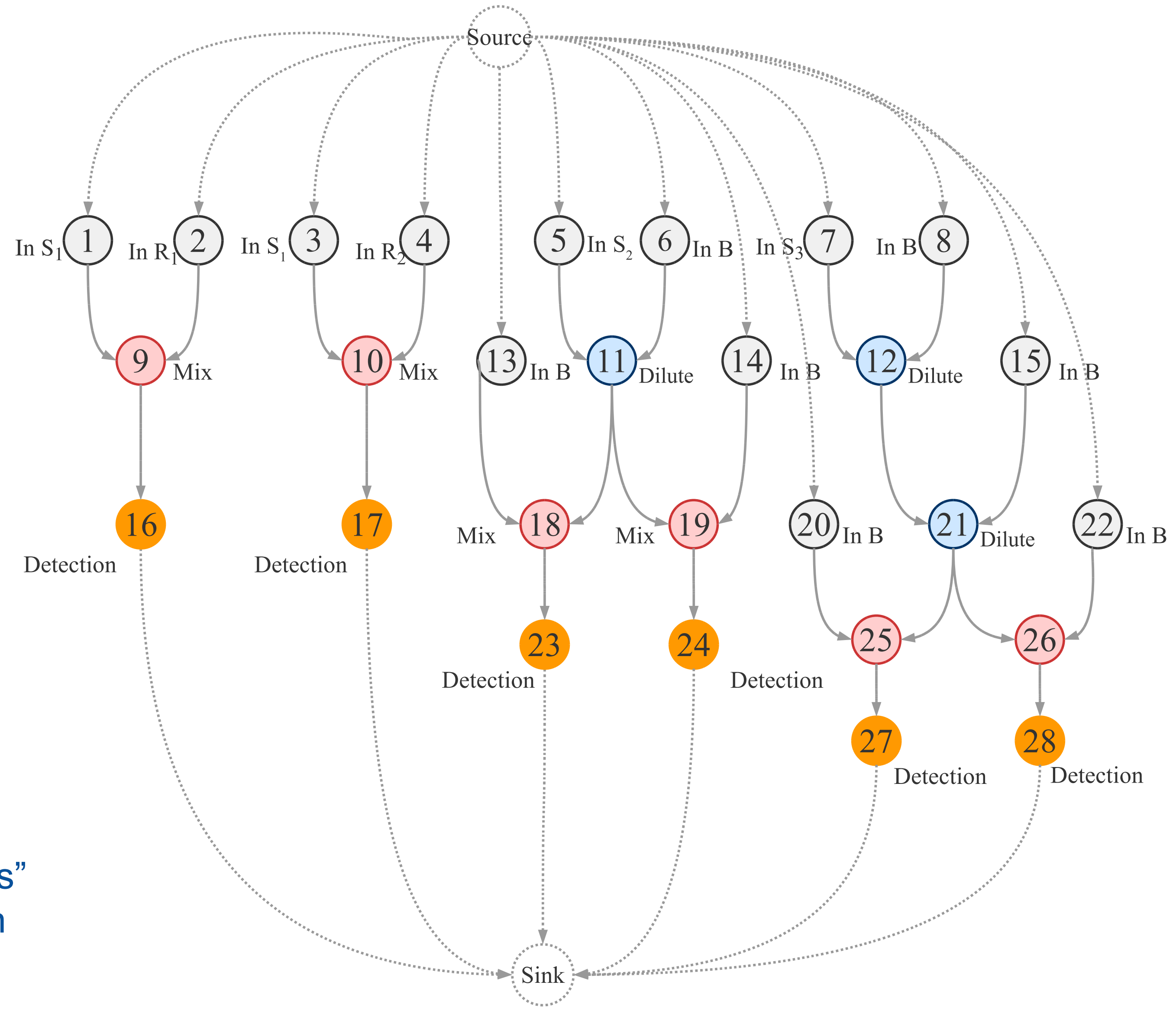


Application mapping: Problem formulation

- **Given**
  - A biochemical application
  - A biochip modeled as a topology graph
  - Characterized component model library

- **Determine**
  - An application mapping, deciding on:
    - **Binding** of operations and edges
    - **Scheduling** of operations and edges
  - **Such that**
    - the application completion time is minimized
    - the dependency, resource and routing constraints are satisfied

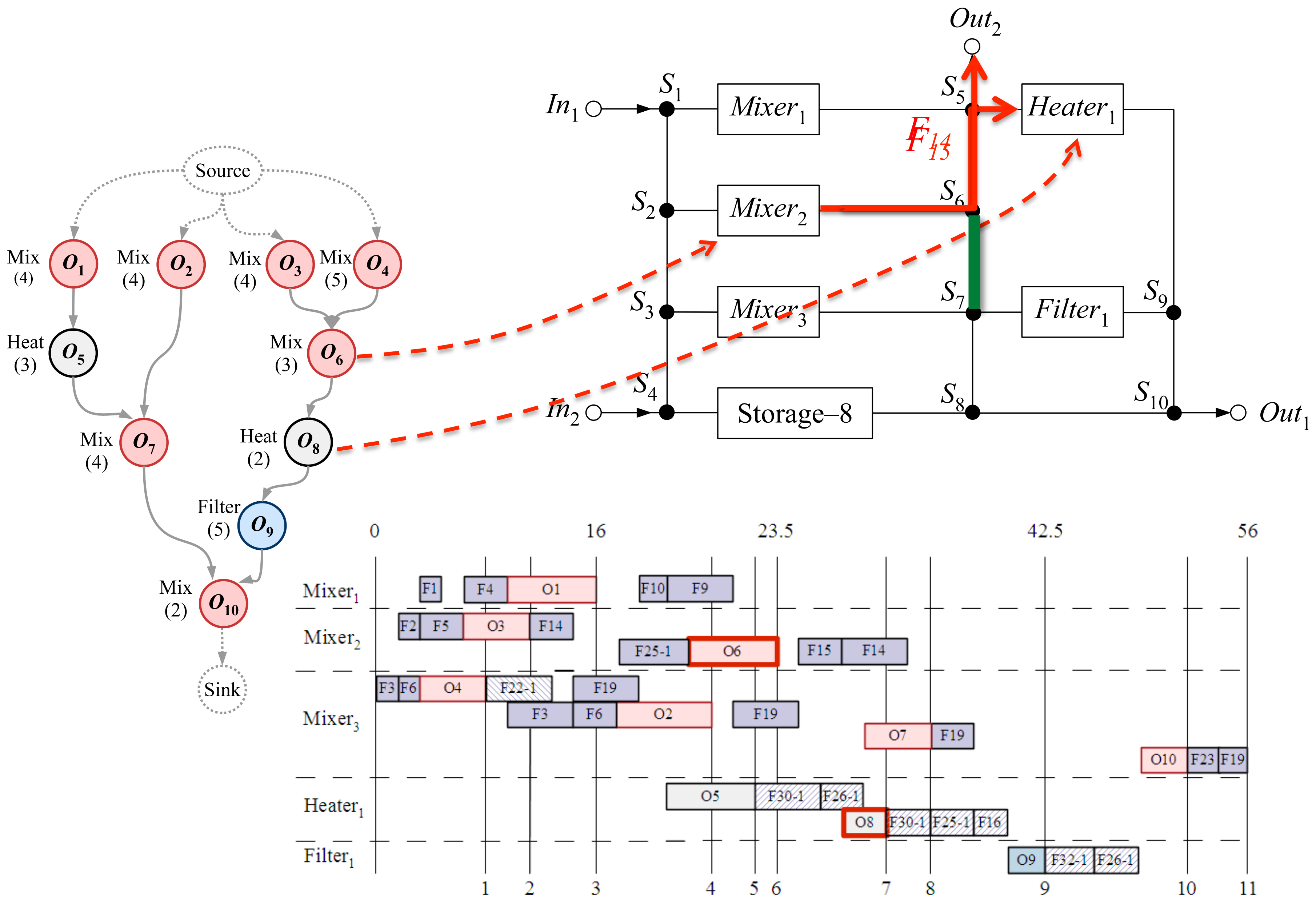
Application model



"in-vitro diagnostics" application

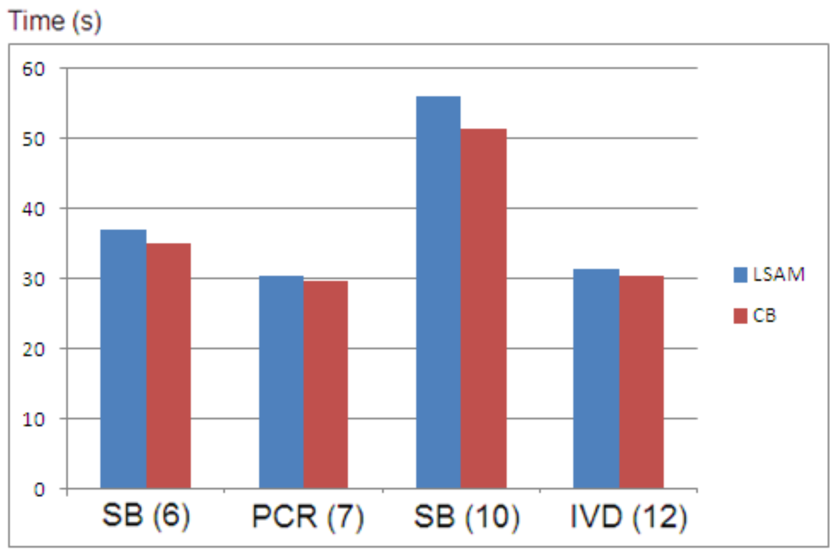
Proposed solution

- List Scheduling-based Application Mapping (LSAM)
  - Binding
  - Scheduling
  - Fluidic routing (contention awareness)
  - Storage (requirement analysis and assignment)
  - Composite route generation


Source
Mix (4) O_1
Mix (4) O_2
Mix (4) O_3
Mix (5) O_4
Heat (3) O_5
Mix (3) O_6
Mix (4) O_7
Heat (2) O_8
Filter (5) O_9
Mix (2) O_{10}
Sink
In_1
In_2
Out_1
Out_2
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10}
$Mixer_1$
$Mixer_2$
$Mixer_3$
$Heater_1$
$Filter_1$
Storage–8
~~F_{14}~~ F_{15}
0 16 23.5 42.5 56
$Mixer_1$: F1 F4 O1 F10 F9
$Mixer_2$: F2 F5 O3 F14 F25-1 O6 F15 F14
$Mixer_3$: F3 F6 O4 F22-1 F19 F3 F6 O2 F19 O7 F19 O10 F23 F19
$Heater_1$: O5 F30-1 F26-1 O8 F30-1 F25-1 F16
$Filter_1$: O9 F32-1 F26-1
1 2 3 4 5 6 7 8 9 10 11

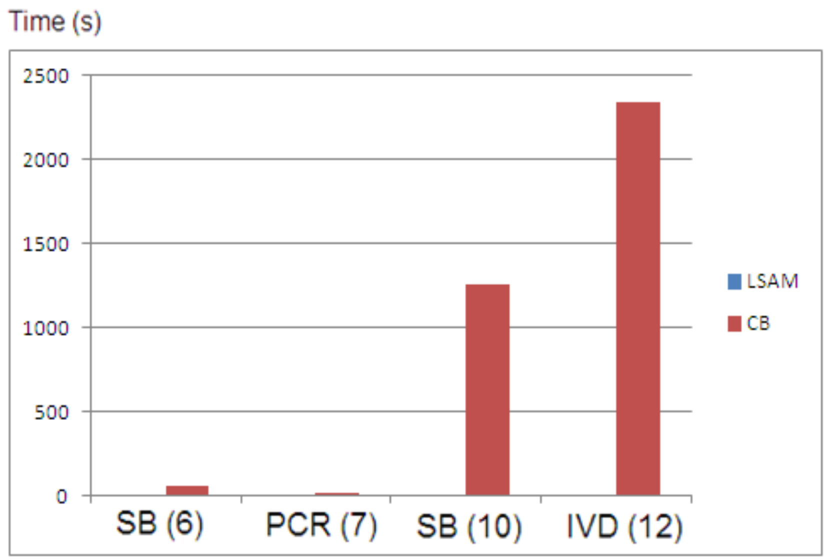
LSAM comparison with optimal

Schedule length



LSAM produces good quality solutions in short time.

Computation time



CB:	Clique based optimal solution [Dinh et al. ASPDAC, 2013]
SB:	Synthetic benchmark
PCR:	Polymerase chain reaction – mixing stage
IVD:	In-vitro diagnostics

Summary and message

- Summary
  - Models for the biochip and the components
  - Top-down physical synthesis and application mapping
    - Design for fault-tolerance
  - Expected to
    - increase the yield
    - facilitate programmability and automation
    - minimize design cycle time
    - enhance chip scalability and throughput
    - play a role in emergence of a large biochip market
- Message
  - Deign complexity is on the rise
  - Top-down CAD tools are needed to support the designer
  - The introduction of redundancy has to be carefully optimized
  - High-level language and tools are needed for programability