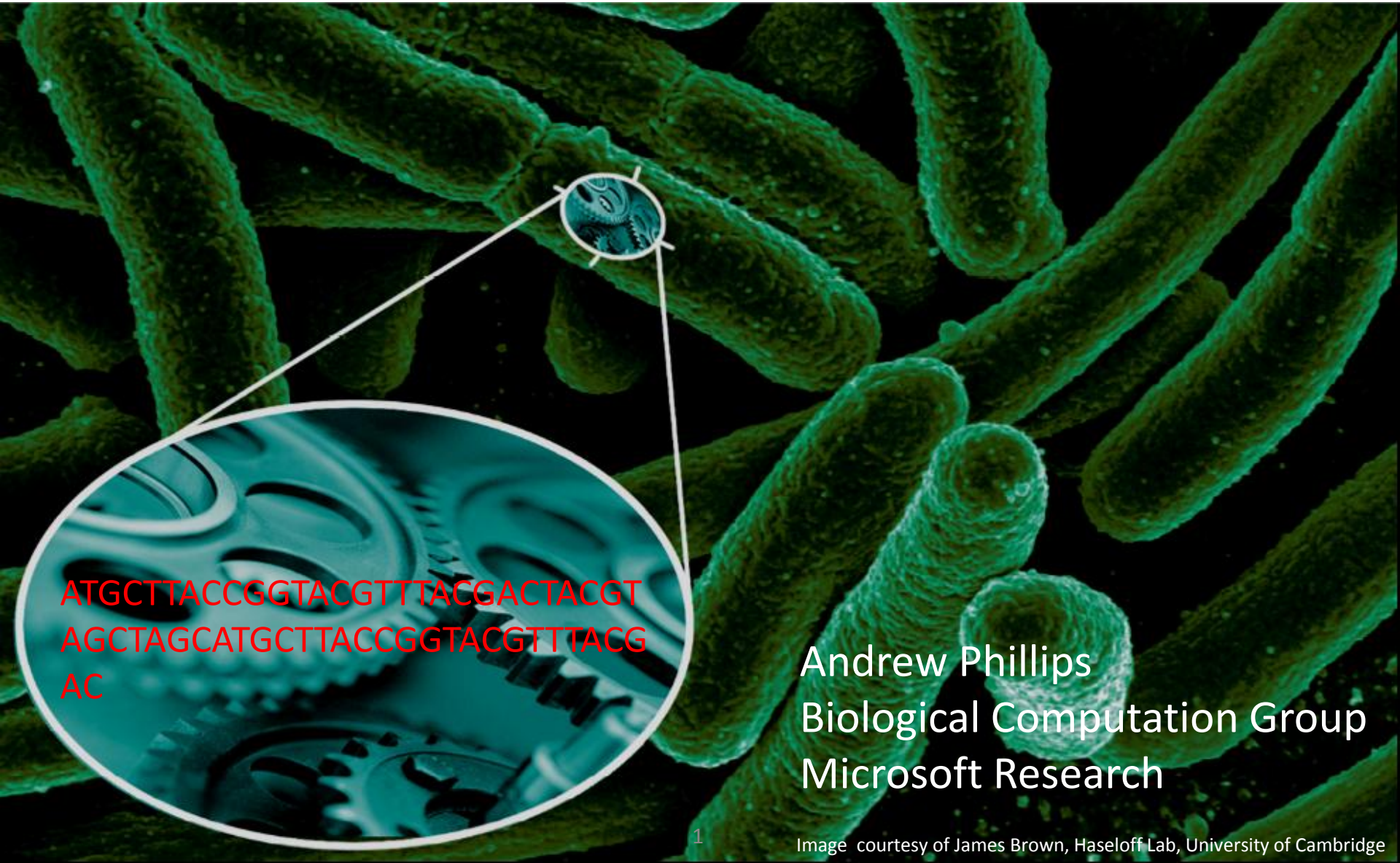


# Programming Cells



ATGCTTACCGGTACGTTTACGACTACGT  
AGCTAGCATGCTTACCGGTACGTTTACG  
AC

Andrew Phillips  
Biological Computation Group  
Microsoft Research

# Potential

## Health

Program cells to control tumours

**JMB**

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®



### Environmentally Controlled Invasion of Cancer Cells by Engineered Bacteria

J. Christopher Anderson<sup>1,3</sup>, Elizabeth J. Clarke<sup>3</sup>, Adam P. Arkin<sup>1,2\*</sup> and Christopher A. Voigt<sup>2,3</sup>



Aerobic Conditions  
Low Cell Density  
OFF

Hypoxia  
High Cell Density  
ON

inv Induction → Invasion

## Energy

Convert CO<sub>2</sub> into fuel



Produce vaccines

Convert sunlight into electricity

nature Vol 440:13 April 2006 doi:10.1038/nature04640

LETTERS

### Production of the antimalarial drug precursor artemisinic acid in engineered yeast

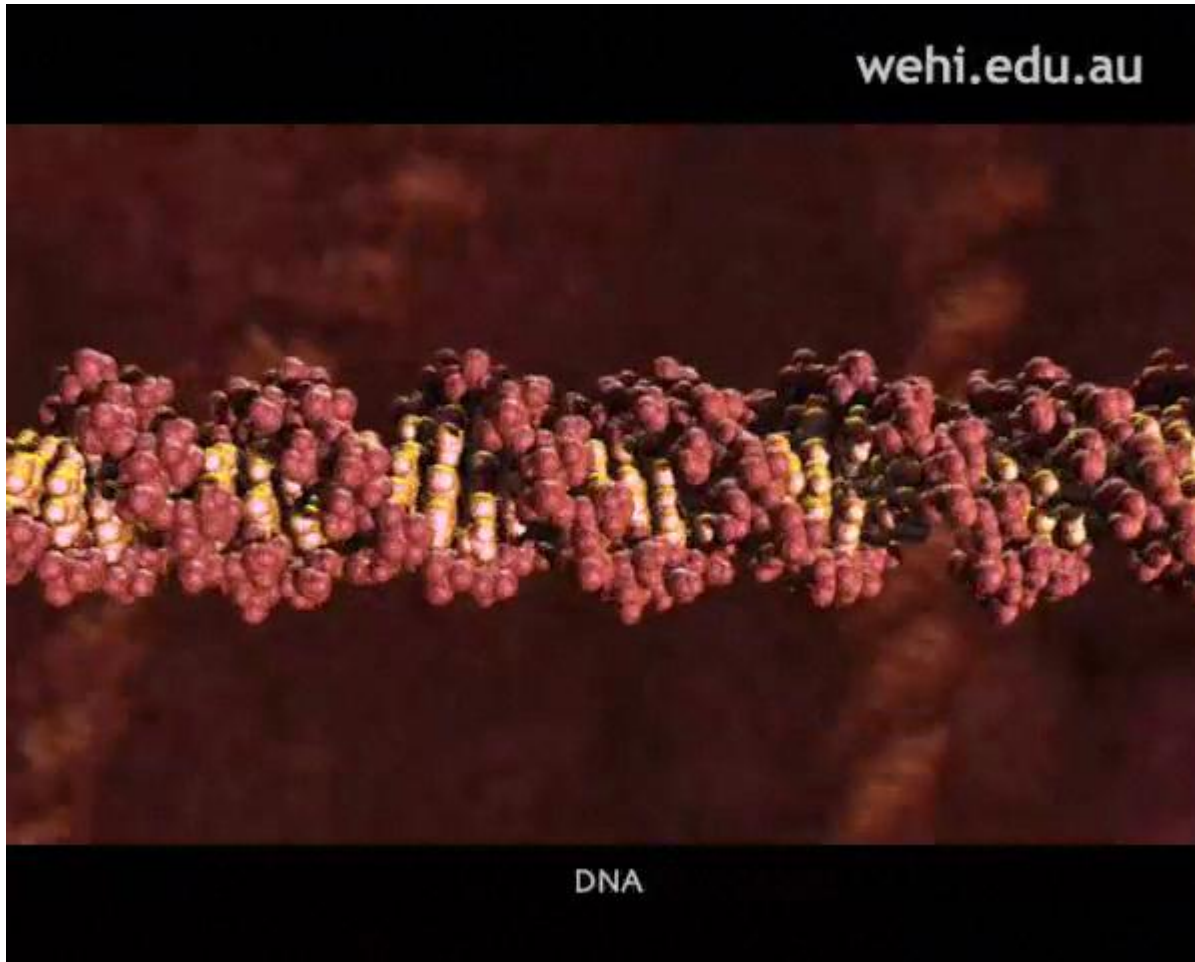
Dae-Kyun Ro<sup>1\*</sup>, Eric M. Paradise<sup>2\*</sup>, Mario Ouellet<sup>1</sup>, Karl J. Fisher<sup>2</sup>, Karyn L. Newman<sup>1</sup>, John M. Ndungu<sup>1</sup>, Kimberly A. Ho<sup>1</sup>, Rachel A. Eachus<sup>1</sup>, Timothy S. Ham<sup>1</sup>, James Kirby<sup>2</sup>, Michelle C. Y. Chang<sup>1</sup>, Sydnor T. Withers<sup>2</sup>, Yoichiro Shiba<sup>2</sup>, Richmond Sarpong<sup>2</sup> & Jay D. Keasling<sup>1,2,4,5</sup>



# Challenges

- Programming cells is hugely difficult
  - Issues of reliability, toxicity, strain on the host cell
- Systems are highly complex
  - Cannot be designed by trial and error
  - Requires use of computer software
- Could software for programming cells one day rival software for programming silicon?

# DNA Structure (A-T,G-C)



CCCAAACAAAACAAAACAA  
GGGTTTTGTTTTGTTTTGTT

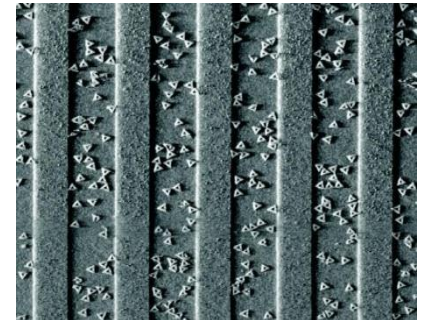
CCCTTTTCTAACTAAACAA  
GGGAAAAGATTTGATTTGTT



# DNA as a Computing Substrate

- Molecular Scale

- Overcome limits to miniaturisation of silicon chips
- 1GB of information in a millionth of a mm<sup>3</sup>
- Can self-assemble
- Clean and cheap to manufacture



Placement and orientation of individual DNA shapes on lithographically patterned surfaces. Nature Nanotechnology 4, 557 - 561 (2009).

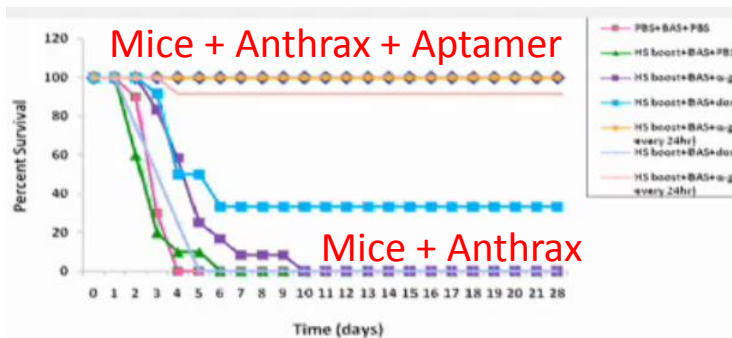
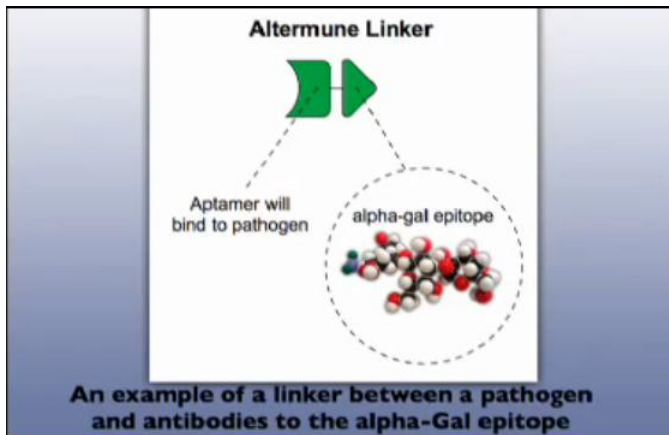
- Programmable

- Interactions are directly controlled by the sequence

- Massively parallel

# DNA Computers Inside Cells

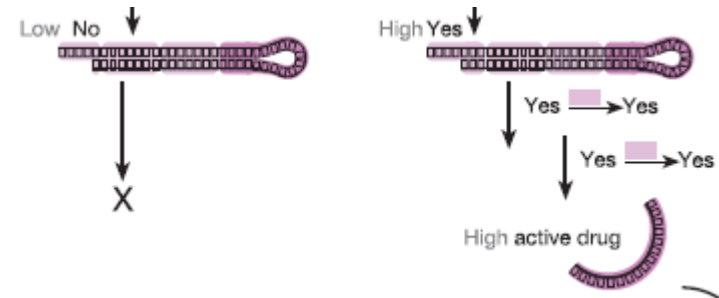
## DNA Drugs (Mullis)



## Programmable Drugs (Shapiro)

### An autonomous molecular computer for logical control of gene expression

Yaakov Benenson<sup>1,2</sup>, Binyamin Gil<sup>2</sup>, Uri Ben-Dor<sup>1</sup>, Rivka Adar<sup>2</sup> & Ehud Shapiro<sup>1,2</sup>



©2004 Nature Publishing Group

# DNA Strand Displacement

Computation solely by complementary base pairs sticking together (T-A and G-C)

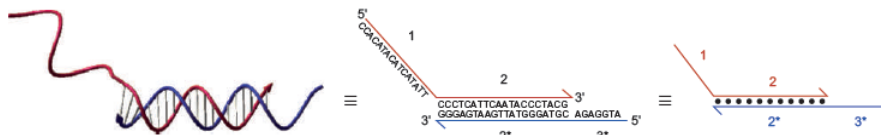


Bernard Yurke

# DNA Strand Displacement

## Dynamic DNA nanotechnology using strand displacement reactions

David Yu Zhang<sup>1</sup> and Georg Seelig<sup>2</sup>

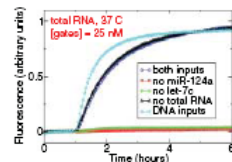
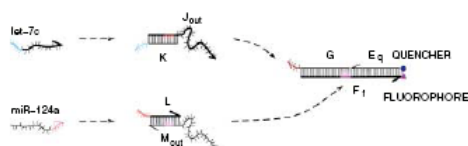


NATURE CHEMISTRY | VOL 3 | FEBRUARY 2011

## Enzyme-Free Nucleic Acid Logic Circuits

Georg Seelig,<sup>1</sup> David Soloveichik,<sup>2</sup> David Yu Zhang,<sup>2</sup> Erik Winfree<sup>2,3\*</sup>

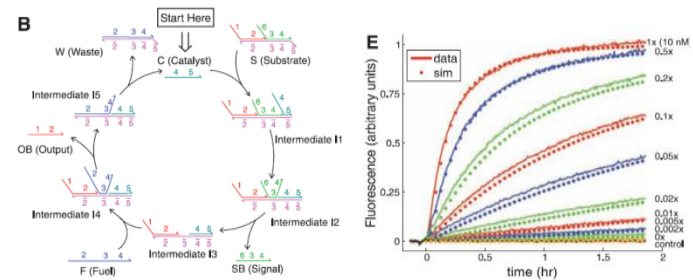
A CIRCUIT DIAGRAM FOR: let-7c AND miR-124a



SCIENCE VOL 314 8 DECEMBER 2006

## Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA

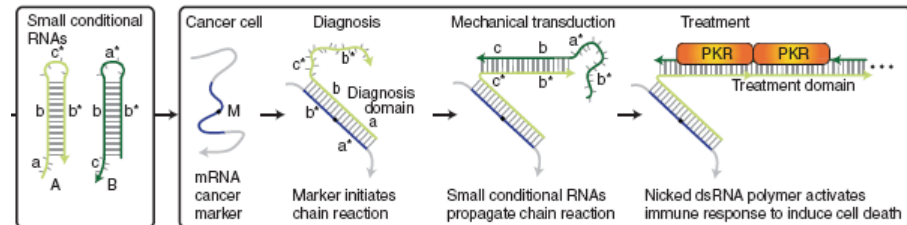
David Yu Zhang,<sup>1†</sup> Andrew J. Turberfield,<sup>2</sup> Bernard Yurke,<sup>3\*</sup> Erik Winfree<sup>1†</sup>



SCIENCE VOL 318 16 NOVEMBER 2007

## Selective cell death mediated by small conditional RNAs

Suvir Venkataraman<sup>a</sup>, Robert M. Dirks<sup>a,b</sup>, Christine T. Ueda<sup>b</sup>, and Niles A. Pierce<sup>a,c,1</sup>



PNAS | September 28, 2010



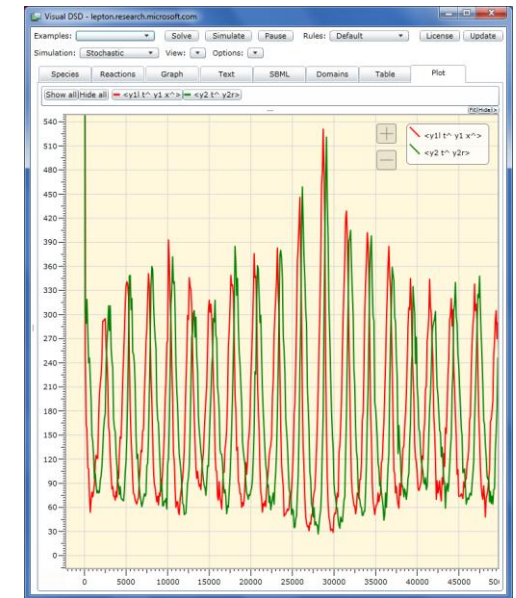
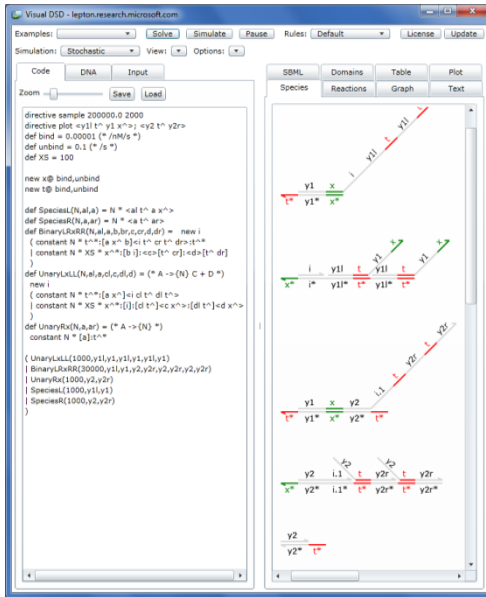
# DNA Strand Displacement Language

## DNA Strand Displacement (DSD) Language

Step 1: Program circuit design

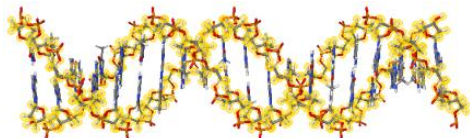
Step 2: Compile circuit behaviour

Step 3: Simulate circuit



Step 4: Compile circuit to DNA or RNA

Step 5: Insert circuit into cells



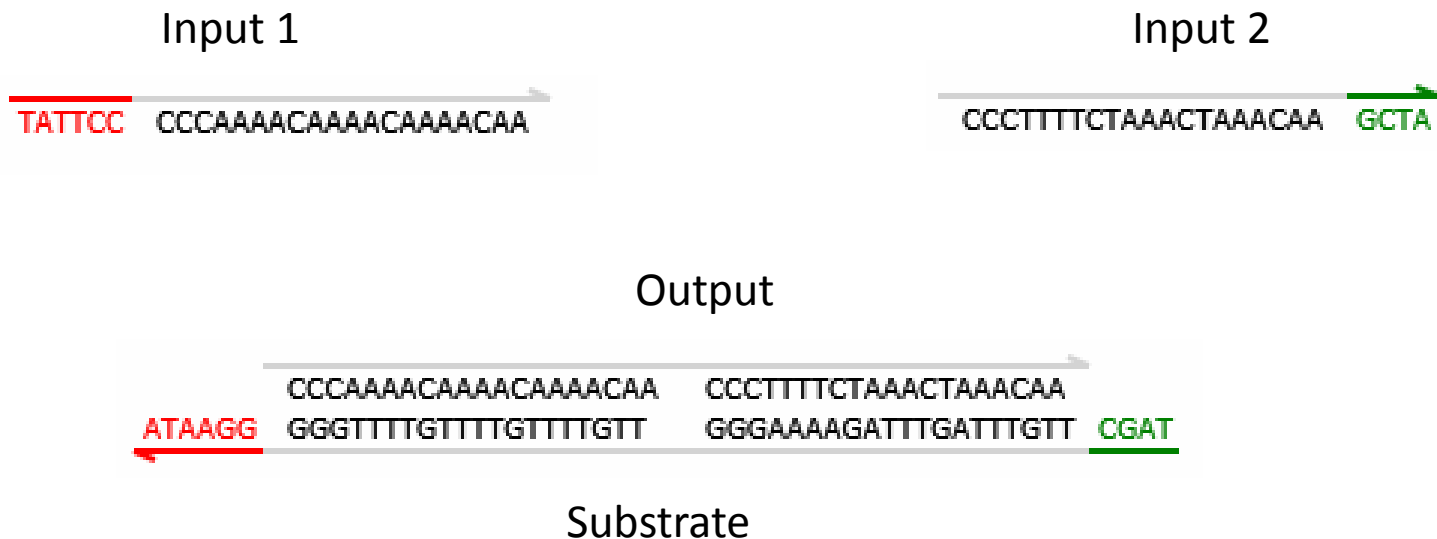
Phillips, Cardelli. Royal Society Interface, 2009

Lakin, Youssef, Cardelli, Phillips. Royal Society Interface, 2011

9 Lakin, Youssef, Polo, Emmott, Phillips. Bioinformatics, 2011

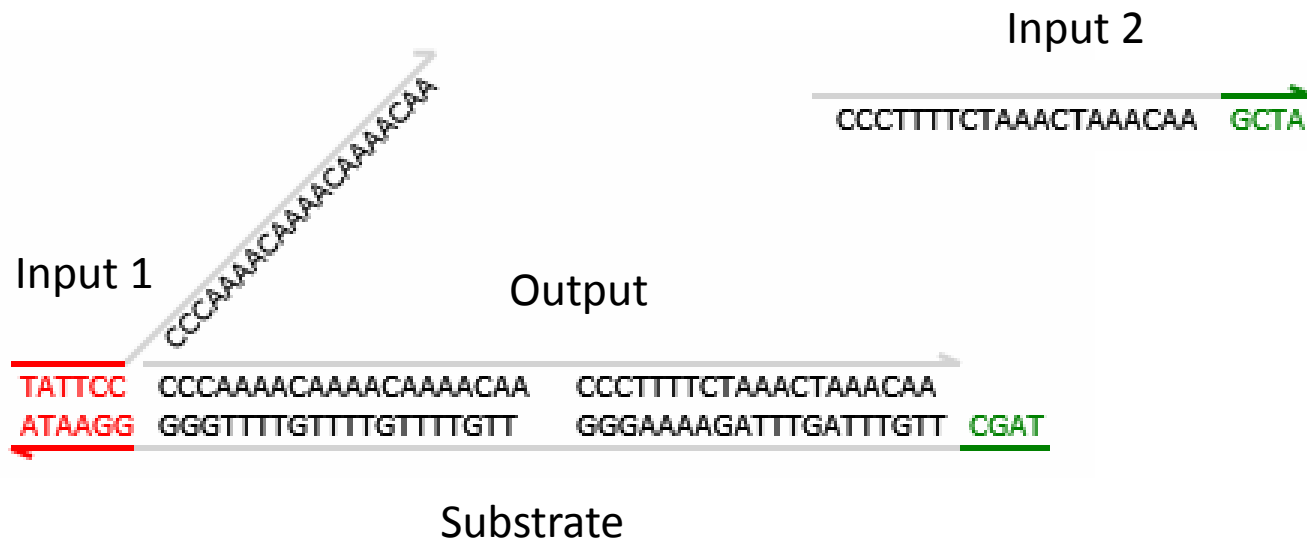
# Strand Displacement Logic Gate

Output = Input1 AND Input2



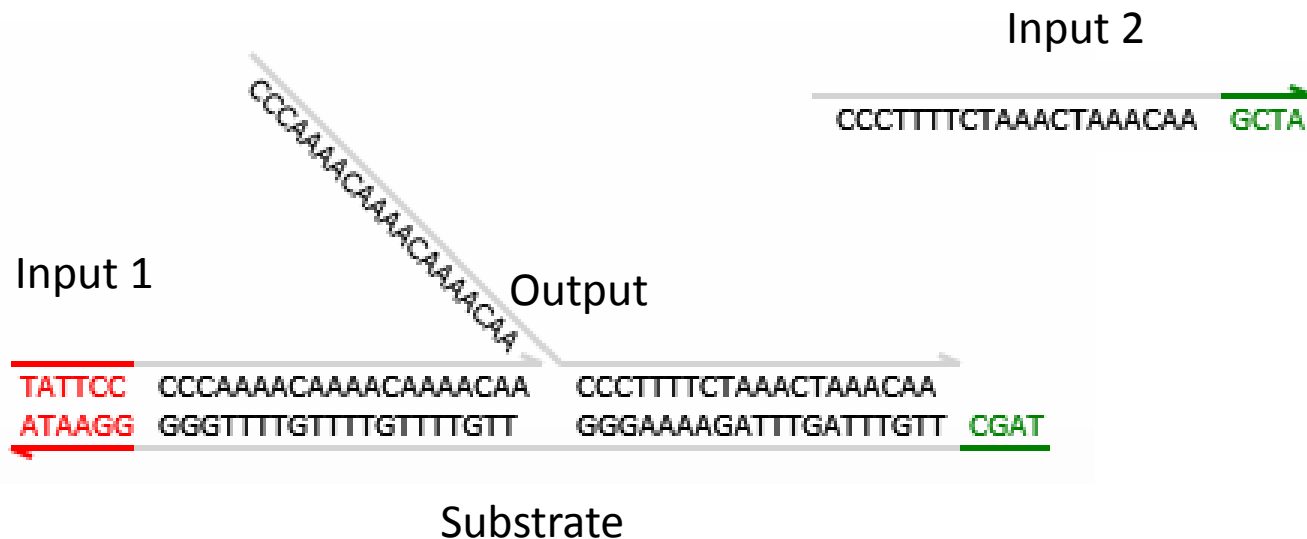
# Strand Displacement Logic Gate

Output = Input1 AND Input2



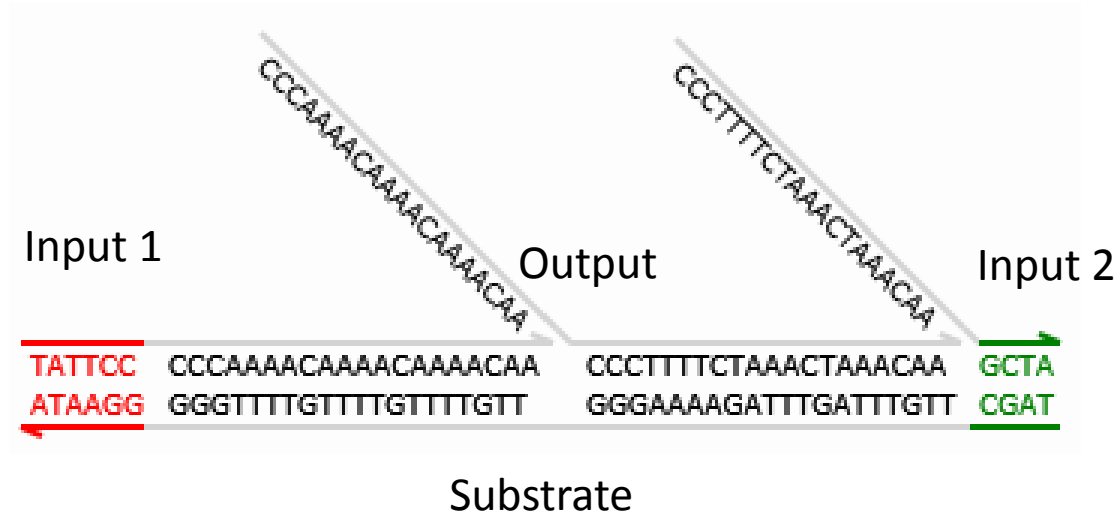
# Strand Displacement Logic Gate

Output = Input1 AND Input2



# Strand Displacement Logic Gate

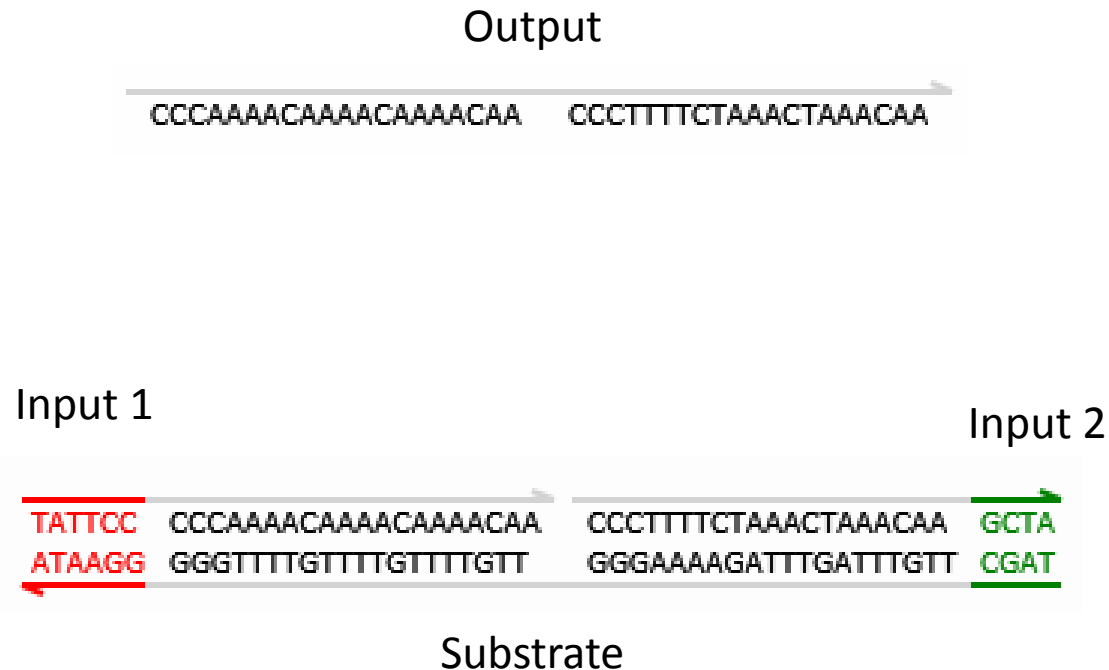
Output = Input1 AND Input2





# Strand Displacement Logic Gate

Output = Input1 AND Input2



# Strand Displacement Logic Gate

Visual DSD - lepton.research.microsoft.com

Examples:  Compile Simulate Analyse Pause Compilation: Default Options: License Update

Simulation: Deterministic View:

Code DNA Input

```
directive sample 1000.0 1000
directive plot <1^ 2>; <3 4^>;

def N1 = 100
def N2 = 100
def N = 1000
def Input1() = <1^ 2>
def Input2() = <3 4^>

def AND() = {1^*}[2 3]{4^*}

( N1*Input1()
| N2*Input2()
| N*AND()
)
```

R Ln 15 Col 2 Ch 2 INS 130%

Compilation Simulation Analysis

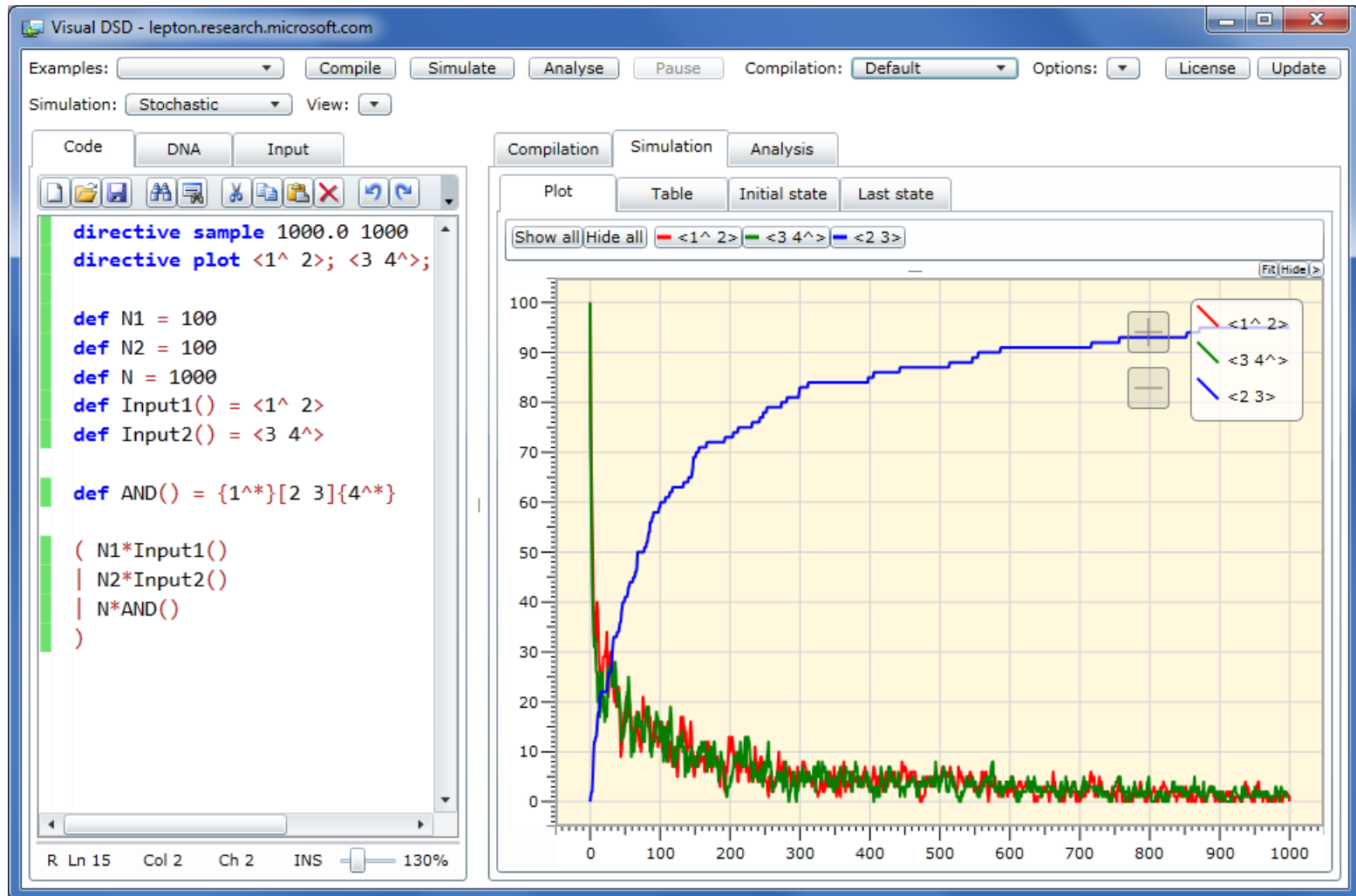
Species Reactions Graph Text Domains SBML

Pan  Zoom  Layout Zoom 81 % Fit Layout

Horizontal  Aspect Ratio  Group Initial Nodes

Diagram illustrating the execution of a DNA logic gate (AND gate) using strand displacement. The diagram shows several DNA strands (represented by colored segments: red, green, black) and their interactions. The strands are arranged in a network, with arrows indicating the direction of strand displacement. The diagram shows the initial state (top row) and the final state (bottom row) of the reaction network, illustrating the execution of the AND gate logic.

# Strand Displacement Logic Gate



# Strand Displacement Logic Gate

The screenshot displays the Visual DSD software interface, which is used for simulating DNA strand displacement. The window title is "Visual DSD - lepton.research.microsoft.com".

**Top Panel:** Includes buttons for "Examples", "Compile", "Simulate", "Analyse", "Pause", "Compilation: Default", "Options", "License", and "Update". Below these are "Simulation: Stochastic" and "View:" dropdown menus.

**Left Panel (Code/DNA/Input):** Shows the input DNA strands:

- Top strand: `CCCAAAACAAAACAAAACAA` (top) and `GGGTTTTGTTTTGTTTTGTT` (bottom). The top strand has a red `ATAAGG` label with a left-pointing arrow, and the bottom strand has a green `CGAT` label with a right-pointing arrow.
- Middle strand: `TATTCC` (top) and `CCCAAAACAAAACAAAACAA` (bottom). The top strand has a red `TATTCC` label with a left-pointing arrow.
- Bottom strand: `CCCTTTTCTAAACTAAACAA` (top) and `GCTA` (bottom). The bottom strand has a green `GCTA` label with a right-pointing arrow.

**Right Panel (Compilation/Simulation/Analysis):** Shows the simulation results as a state transition graph. The graph has three main nodes representing different states of the DNA strands:

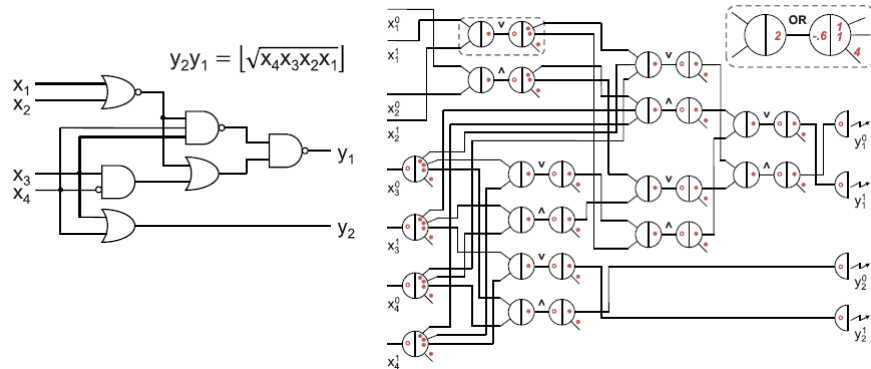
- Top Node:** Shows the initial state with the three input strands as described above.
- Bottom Node:** Shows a state where the top strand is `CCCAAAACAAAACAAAACAA` and the bottom strand is `CCCTTTTCTAAACTAAACAA`. The top strand has a red `TATTCC` label with a left-pointing arrow, and the bottom strand has a green `GCTA` label with a right-pointing arrow.
- Middle Node:** Shows a state where the top strand is `CCCTTTTCTAAACTAAACAA` and the bottom strand is `GGGAAAAGATTGGATTGGTT`. The top strand has a red `ATAAGG` label with a left-pointing arrow, and the bottom strand has a green `CGAT` label with a right-pointing arrow.

Arrows indicate transitions between these states, representing the strand displacement reactions.

# Large-Scale Logic Circuits

## Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

Lulu Qian<sup>1</sup> and Erik Winfree<sup>1,2,3\*</sup>

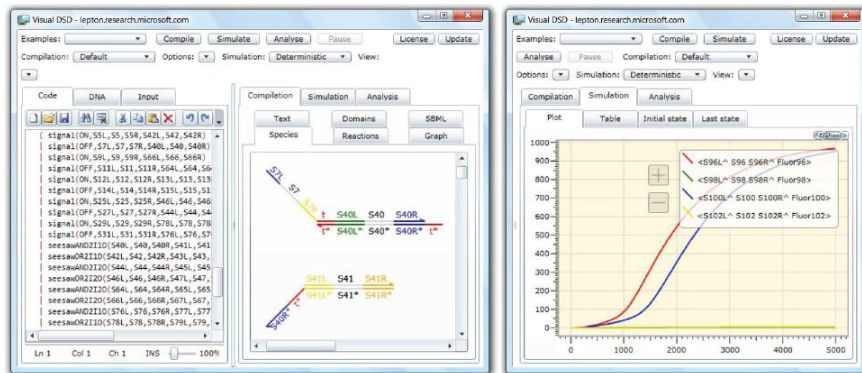


## Scaling Up DNA Computation

John H. Reif

“In addition to biochemistry laboratory techniques, computer science techniques were essential.”

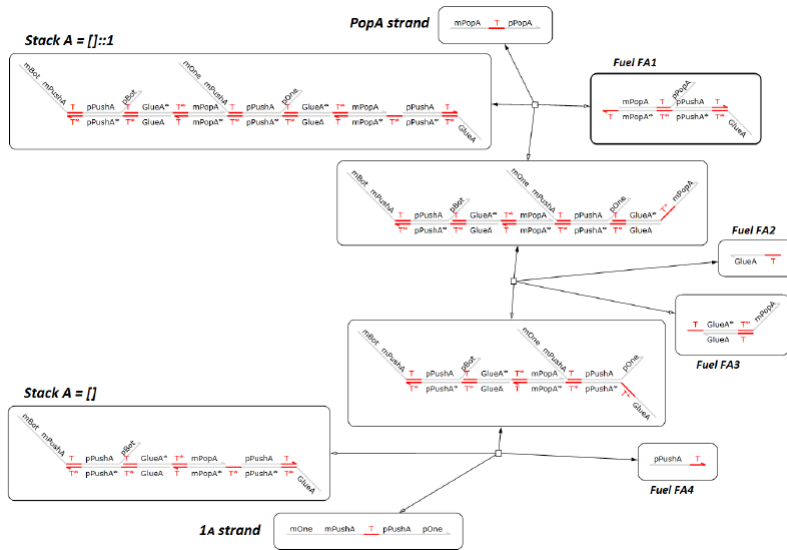
“Computer simulations of seesaw gate circuitry optimized the design and correlated experimental data.”



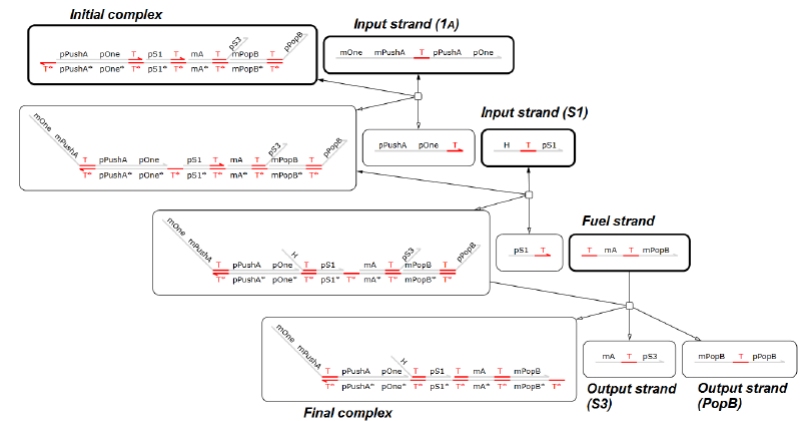


# Turing-Powerful Circuits

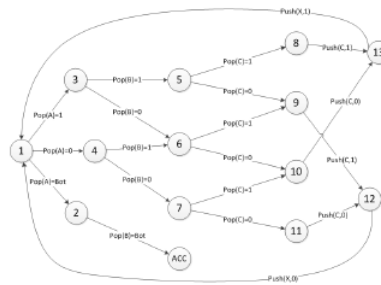
## Encoding a Stack



## Encoding state transitions



## Model-Checking a DNA Ripple Carry Adder

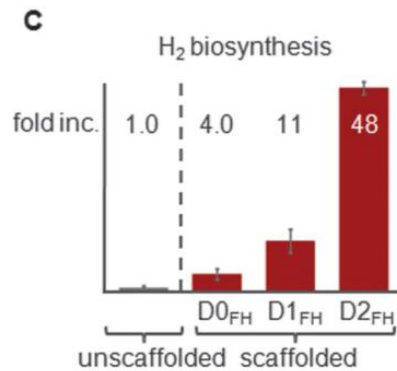
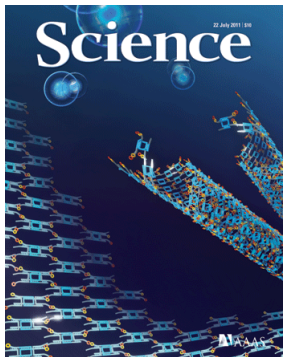


Input A		Input B		Output X		Output C		Result
MSB	LSB	MSB	LSB	LSB	MSB	Value	Value	Value
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	1
0	0	1	0	2	0	1	0	2
0	0	1	1	3	1	1	0	3
0	1	1	0	0	0	1	0	1
0	1	1	0	1	1	0	1	2
0	1	1	1	0	2	1	1	0
0	1	1	1	1	3	0	0	1
1	0	2	0	0	0	0	1	0
1	0	2	0	1	1	1	1	0
1	0	2	1	0	2	0	0	1
1	0	2	1	1	3	1	0	1
1	1	3	0	0	0	1	1	0
1	1	3	0	1	1	0	0	1
1	1	3	1	0	2	1	0	1
1	1	3	1	1	3	0	1	1

# Localised Circuits

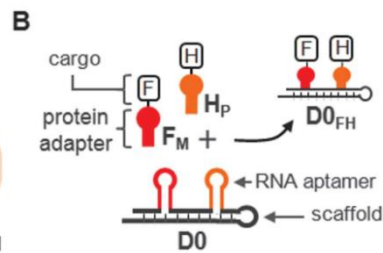
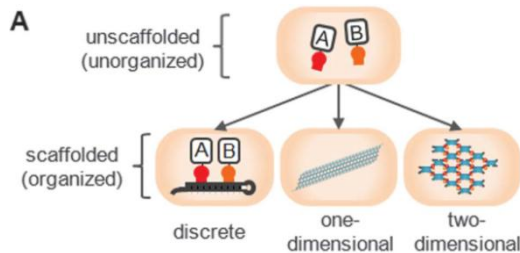
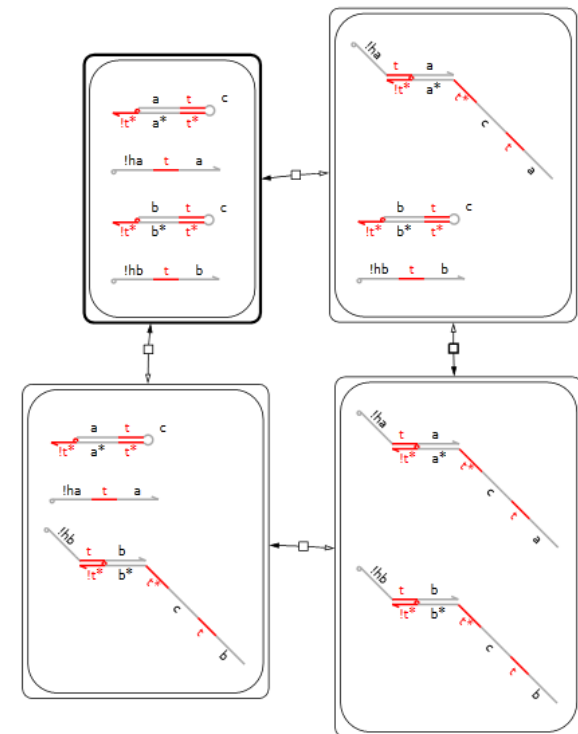
## Organization of Intracellular Reactions with Rationally Designed RNA Assemblies

Camille J. Delebecque,<sup>1,2,3,4</sup> Ariel B. Lindner,<sup>3,4\*</sup> Pamela A. Silver,<sup>1,2\*</sup> Faisal A. Aldaye<sup>1,2</sup>



## Hairpins tethered to origami

- Increased speed
- Reduced interference



# Programming Living Cells

The “software” of a cell: DNA  $\rightarrow$  RNA  $\rightarrow$  Protein



DNA transcription (real time)



Messenger RNA translation

*Information storage and processing within the cell is more efficient by many orders of magnitude than electronic digital computation, with respect to both information density and energy consumption.*

# DNA can function across species

A “multi-platform” code



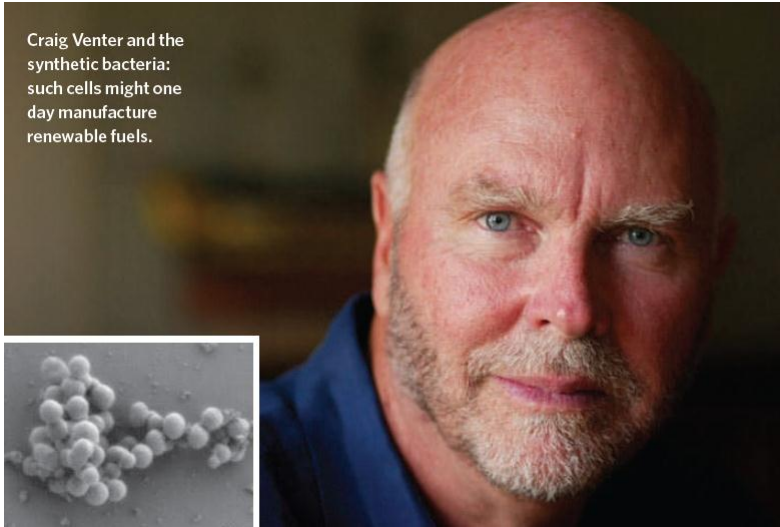
Glowing jellyfish



Glowing bacteria

Moving DNA from one organism to another

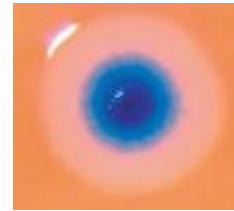
# DNA can completely reprogram a cell



## Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome

Scienceexpress / [www.sciencexpress.org](http://www.sciencexpress.org) / 20 May 2010

*M. mycoides*



Extraction



Sequencing



...GTTTCTCCATACCCGTTTTTTTTGGGCTAGC...

Synthesis



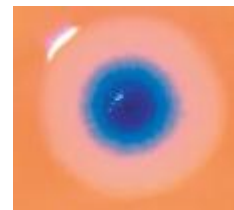
Insertion



*M. capricolum*



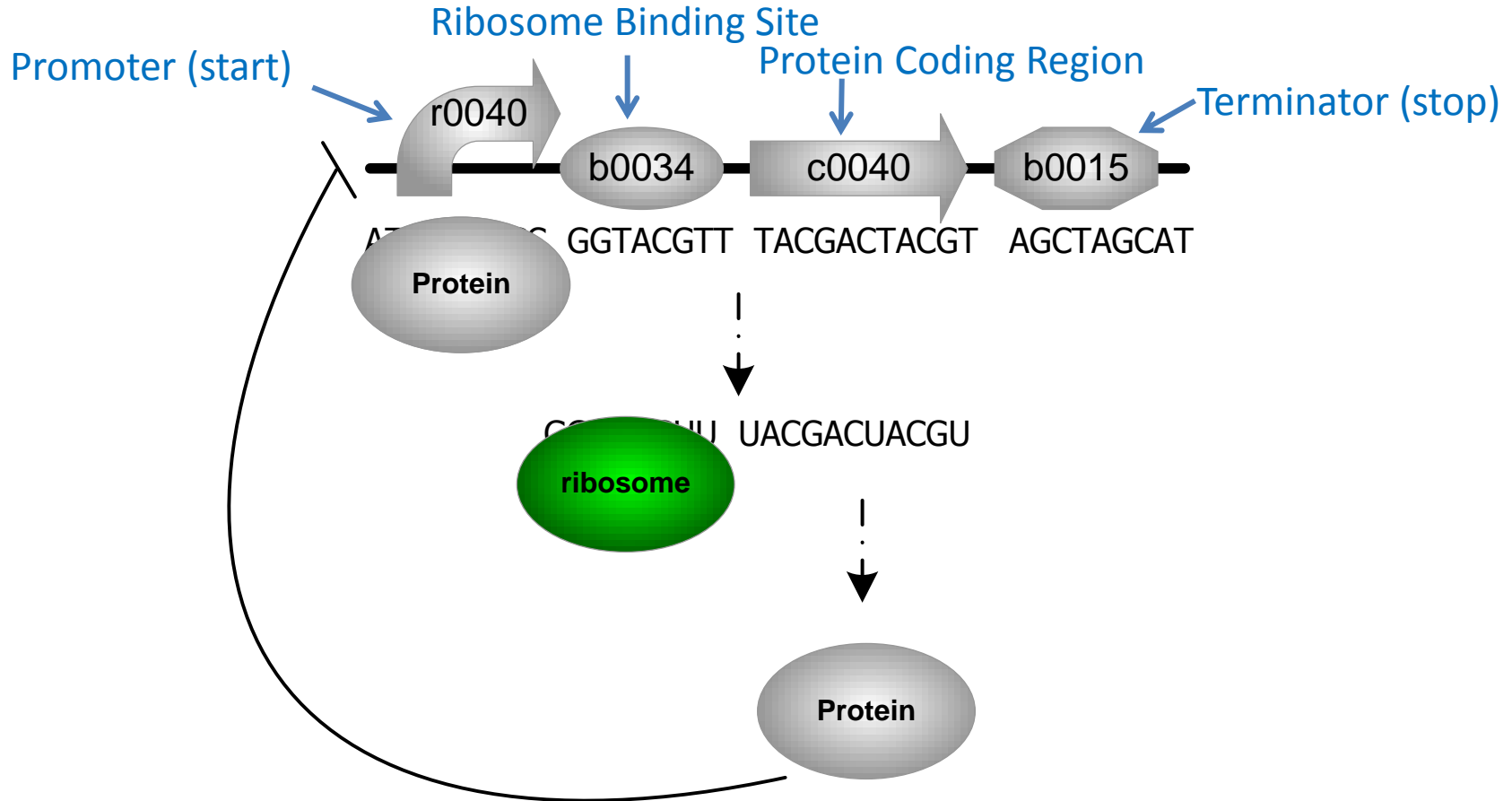
Transformation





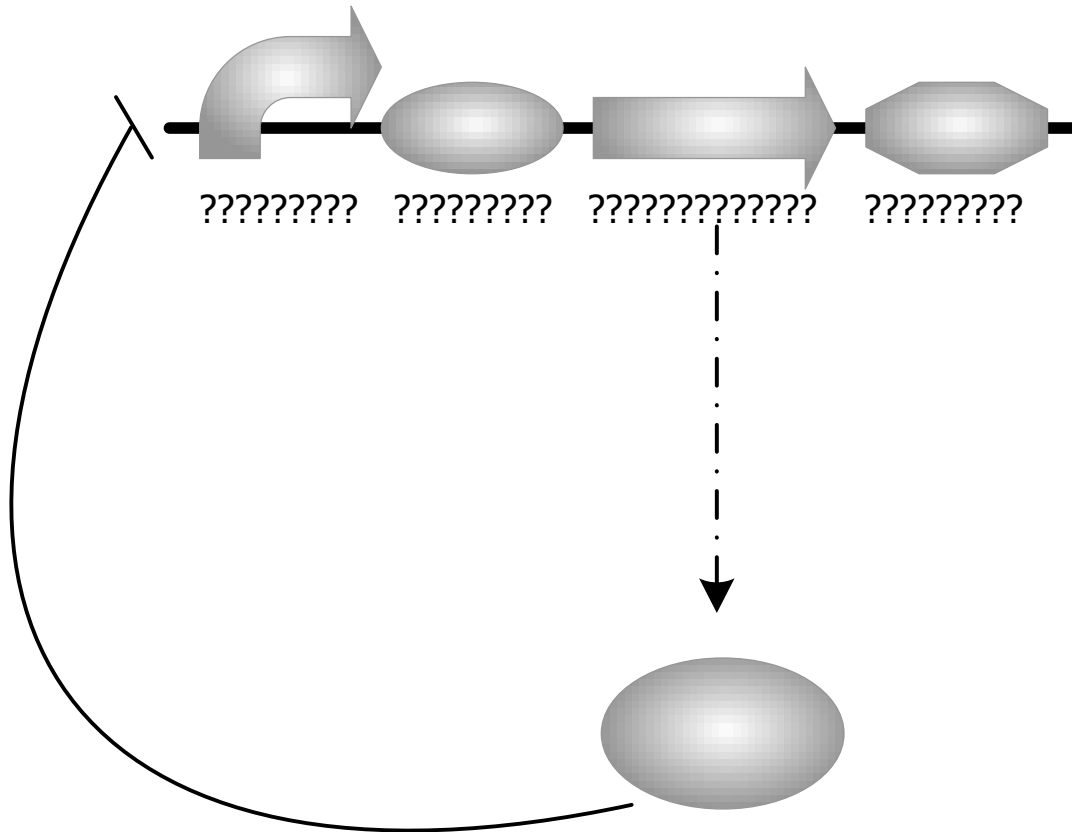
# Low-Level DNA Language

## A simplified view of DNA instructions



# High-Level DNA Language

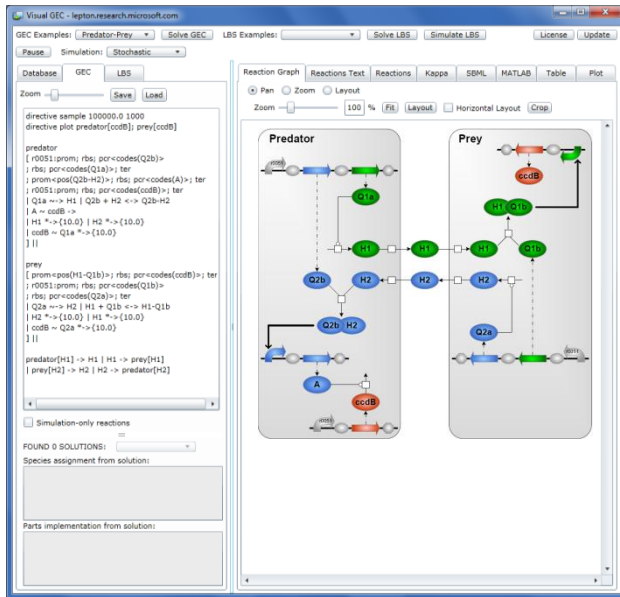
Given a design, automatically determine the DNA



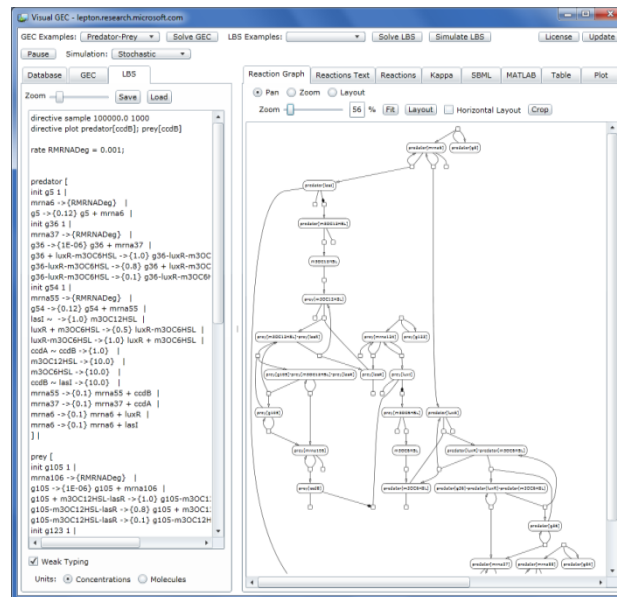
# A language for programming cells

## Genetic Engineering of Cells (GEC)

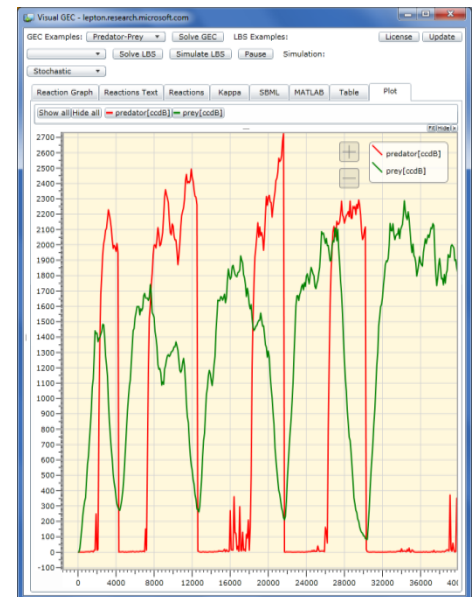
Step 1: Program device design



Step 2: Compile device behaviour



Step 3: Simulate device



Step 4: Compile device to DNA

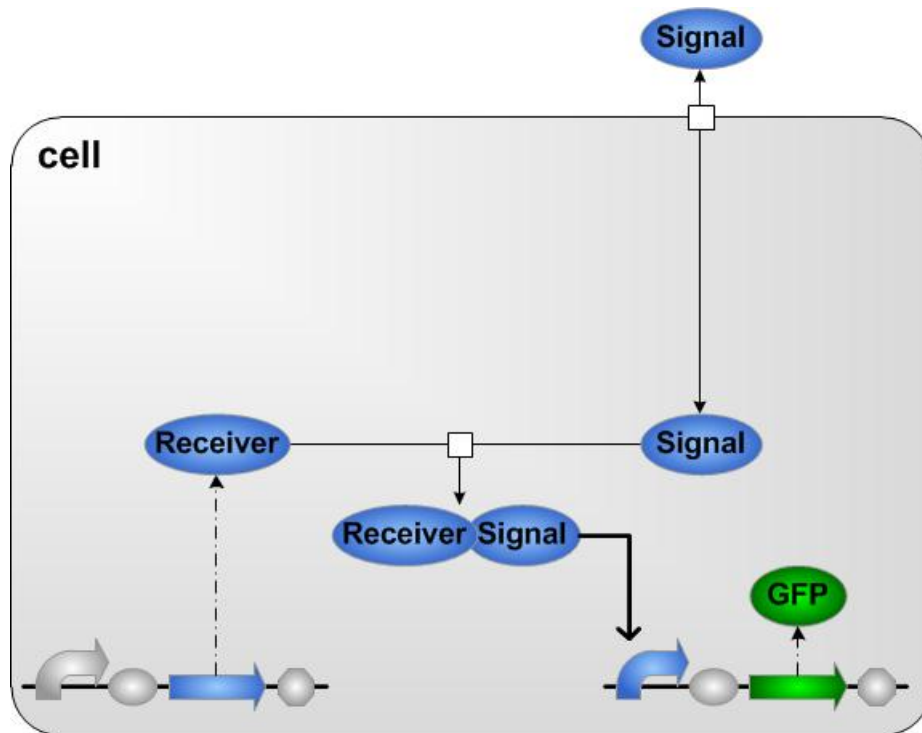


Step 5: Insert DNA into cells



# Programming a receiver device

Signal crosses cell wall and binds to Receiver



```
cell
[ prom; rbs; pcr<codes(Receiver)>; ter
| prom<pos(Receiver-Signal)>;
  rbs; pcr<codes(gfp)>; ter
| Receiver + Signal -> Receiver-Signal
| Receiver-Signal -> Receiver + Signal
]
| Signal -> cell[Signal]
| cell[Signal] -> Signal
| initPop Signal 100.0
```

# Programming a receiver device

The screenshot displays the Visual GEC software interface, which is used for programming and simulating receiver devices. The window title is "Visual GEC - localhost".

**Top Panel:** Contains dropdown menus for "GEC Examples:" and "LBS Examples:", and buttons for "Solve GEC", "Solve LBS", "Simulate LBS", "Pause", "License", and "Update".

**Simulation Panel:** Shows "Simulation: Deterministic" and tabs for "Database", "GEC", and "LBS".

**Code Editor (Left):** Displays the following GEC code:

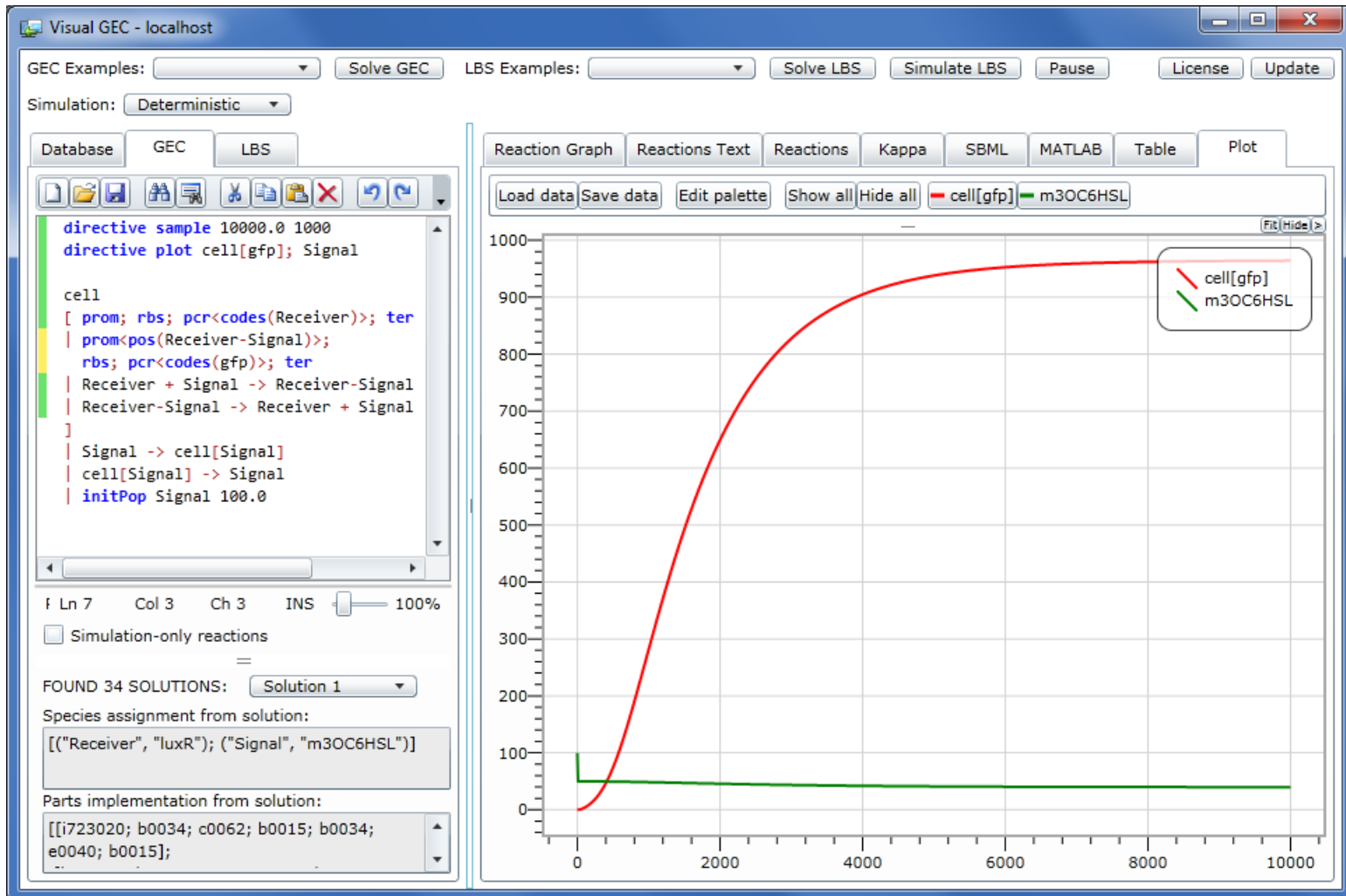
```
directive sample 10000.0 1000
directive plot cell[gfp]; Signal

cell
[ prom; rbs; pcr<codes(Receiver)>; ter
| prom<pos(Receiver-Signal)>;
| rbs; pcr<codes(gfp)>; ter
| Receiver + Signal -> Receiver-Signal
| Receiver-Signal -> Receiver + Signal
]
| Signal -> cell[Signal]
| cell[Signal] -> Signal
| initPop Signal 100.0
```

**Reaction Graph (Right):** Shows a complex network of nodes and edges representing the biochemical reactions. Nodes include "cell[mrna385]", "cell[g252]", "cell[luxR]", "cell[luxR]-cell[m30C6HSL]", "cell[g254]-cell[luxR]-cell[m30C6HSL]", "cell[g254]", "cell[m30C6HSL]", "m30C6HSL", "cell[mrna385]", and "cell[gfp]".

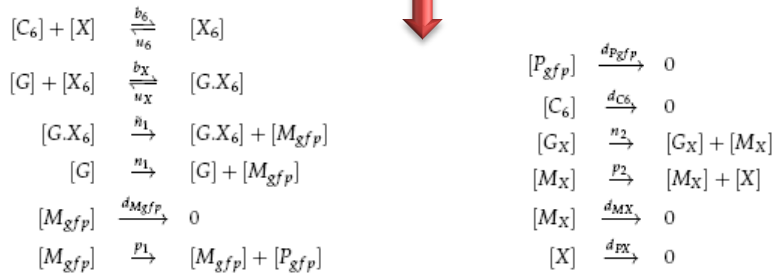
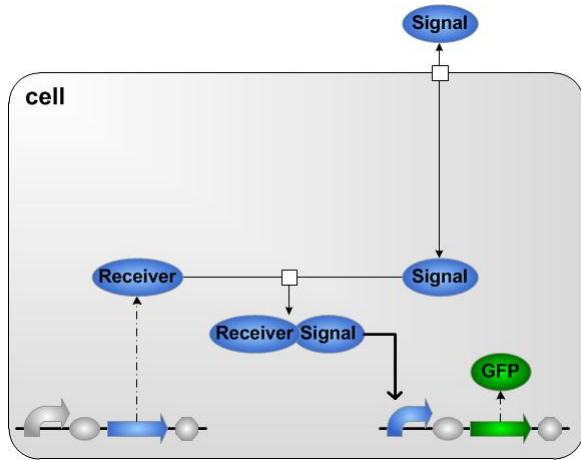
**Bottom Panel:** Shows simulation options and results. It includes a "Simulation-only reactions" checkbox, a "FOUND 34 SOLUTIONS:" section with "Solution 1" selected, and a "Species assignment from solution:" section with the assignment: `[("Receiver", "luxR"); ("Signal", "m30C6HSL")]`. Below this is the "Parts implementation from solution:" section with the implementation: `[["i723020; b0034; c0062; b0015; b0034; e0040; b0015];`

# Programming a receiver device



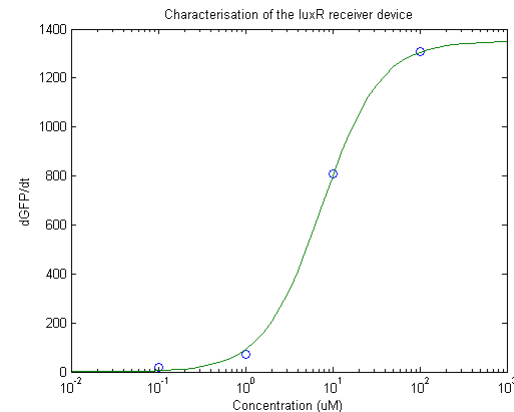
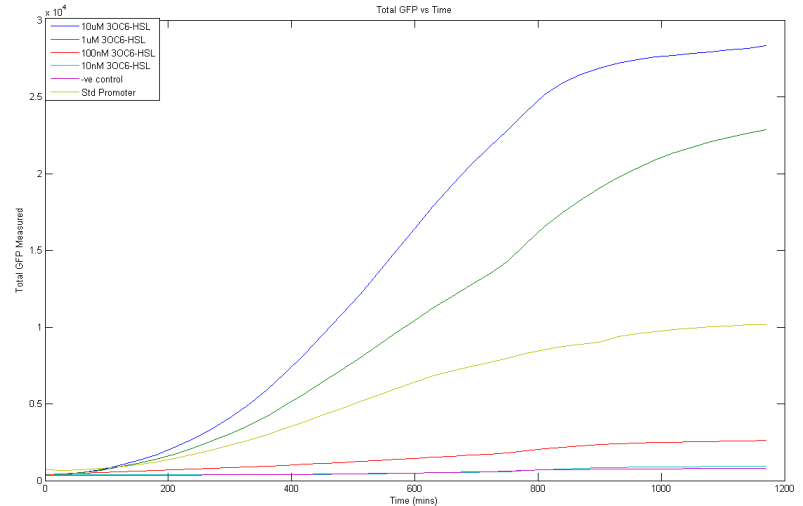
# Characterising a receiver device

## Model Simulation



$$\begin{aligned}
 \frac{d[C_6]}{dt} &= -d_{C_6}[C_6] \\
 \frac{d[P_{GFP}]}{dt} &= \alpha \left( \frac{[C_6] + \epsilon_1 k}{k + [C_6]} \right) - d_{P_{GFP}}[P_{GFP}]
 \end{aligned}
 \quad
 \alpha = \frac{p_1 n_1}{d_{M_{GFP}}}, k = \frac{u_X u_6}{b_X b_6 X_0}$$

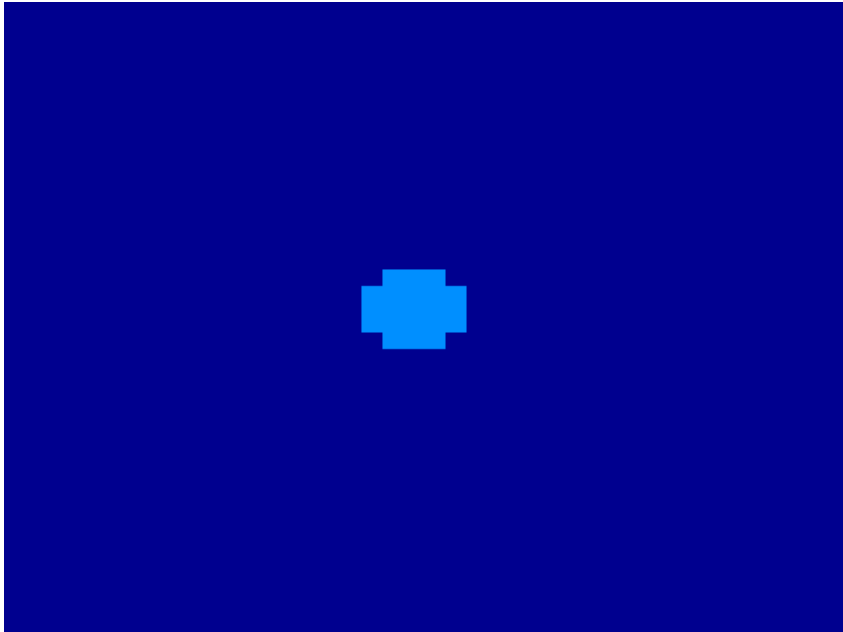
## Experimental Data



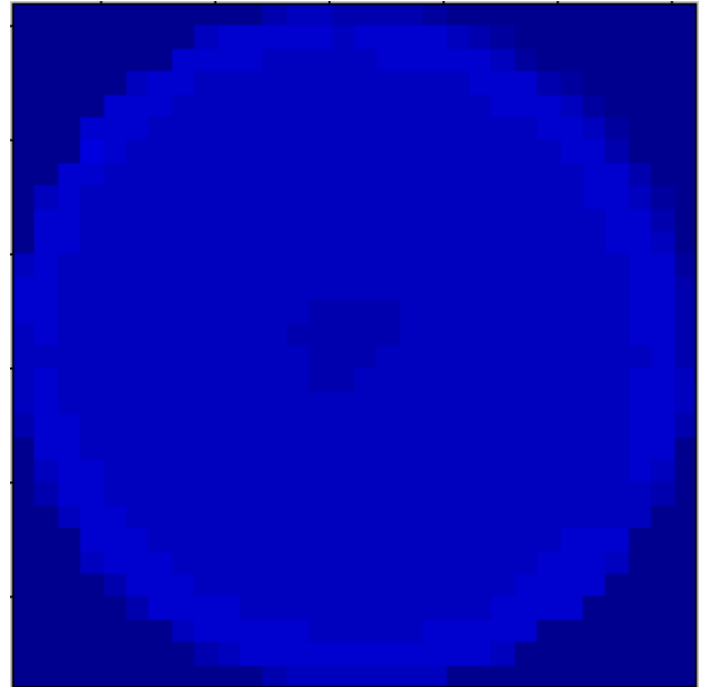


# Spatial Receiver Device

Model Simulation

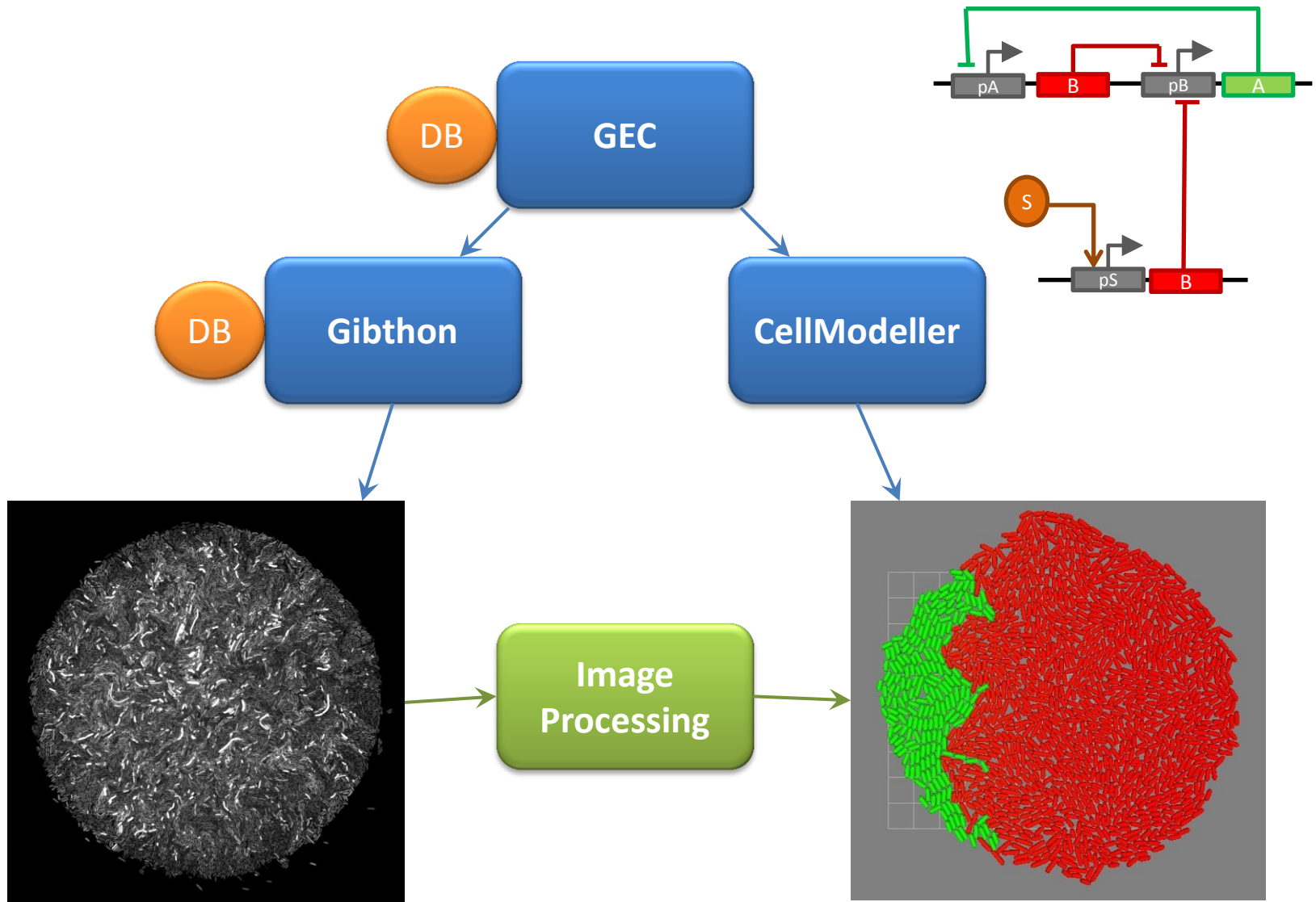


Experimental Data

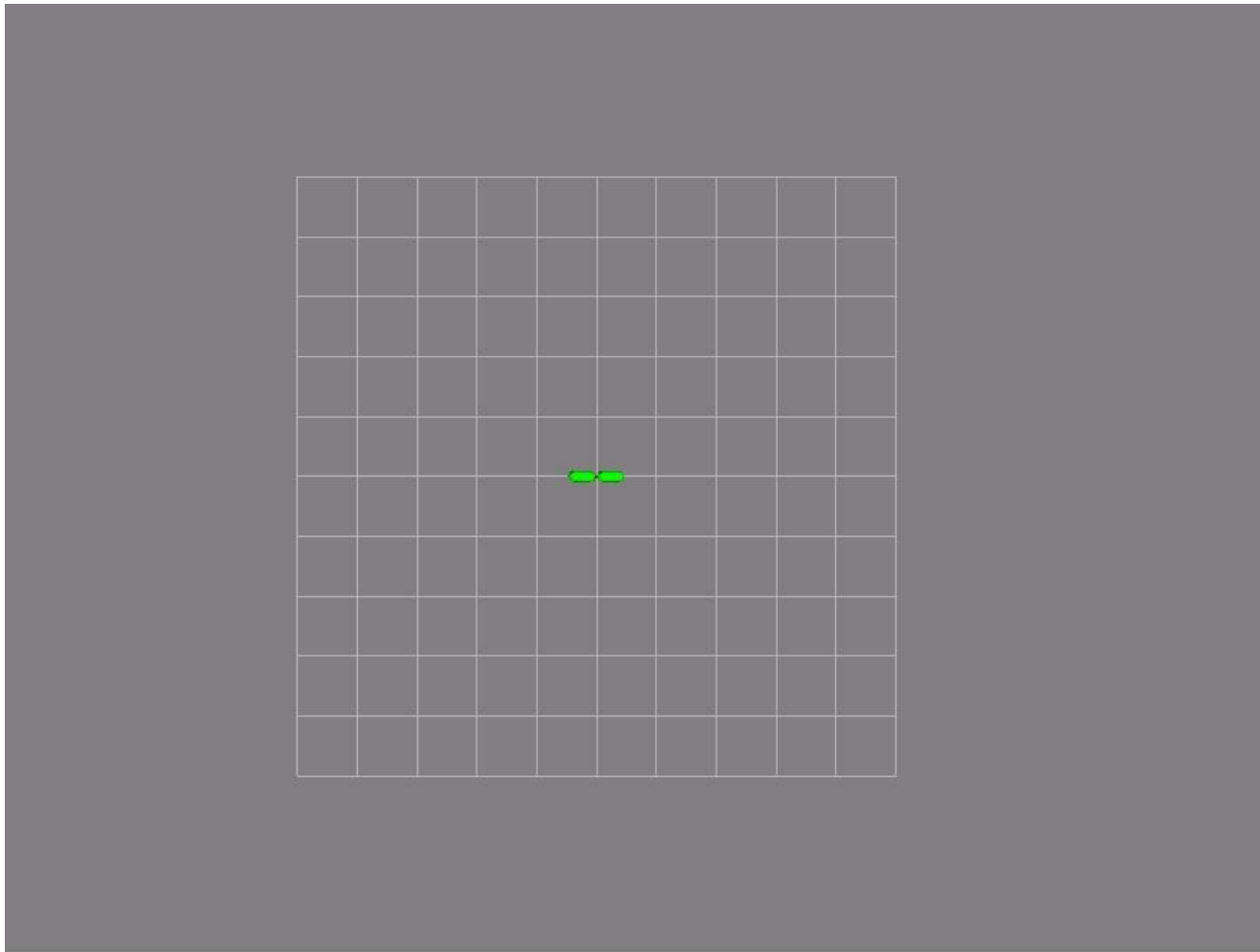
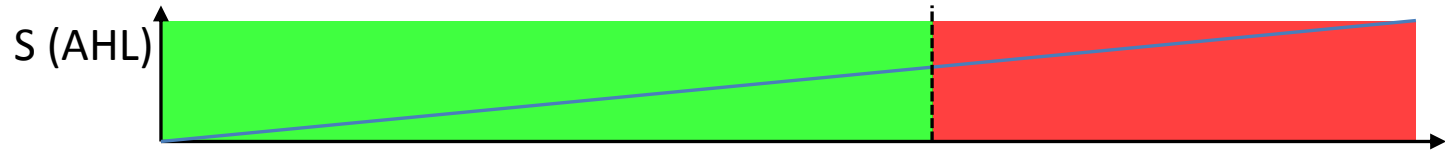




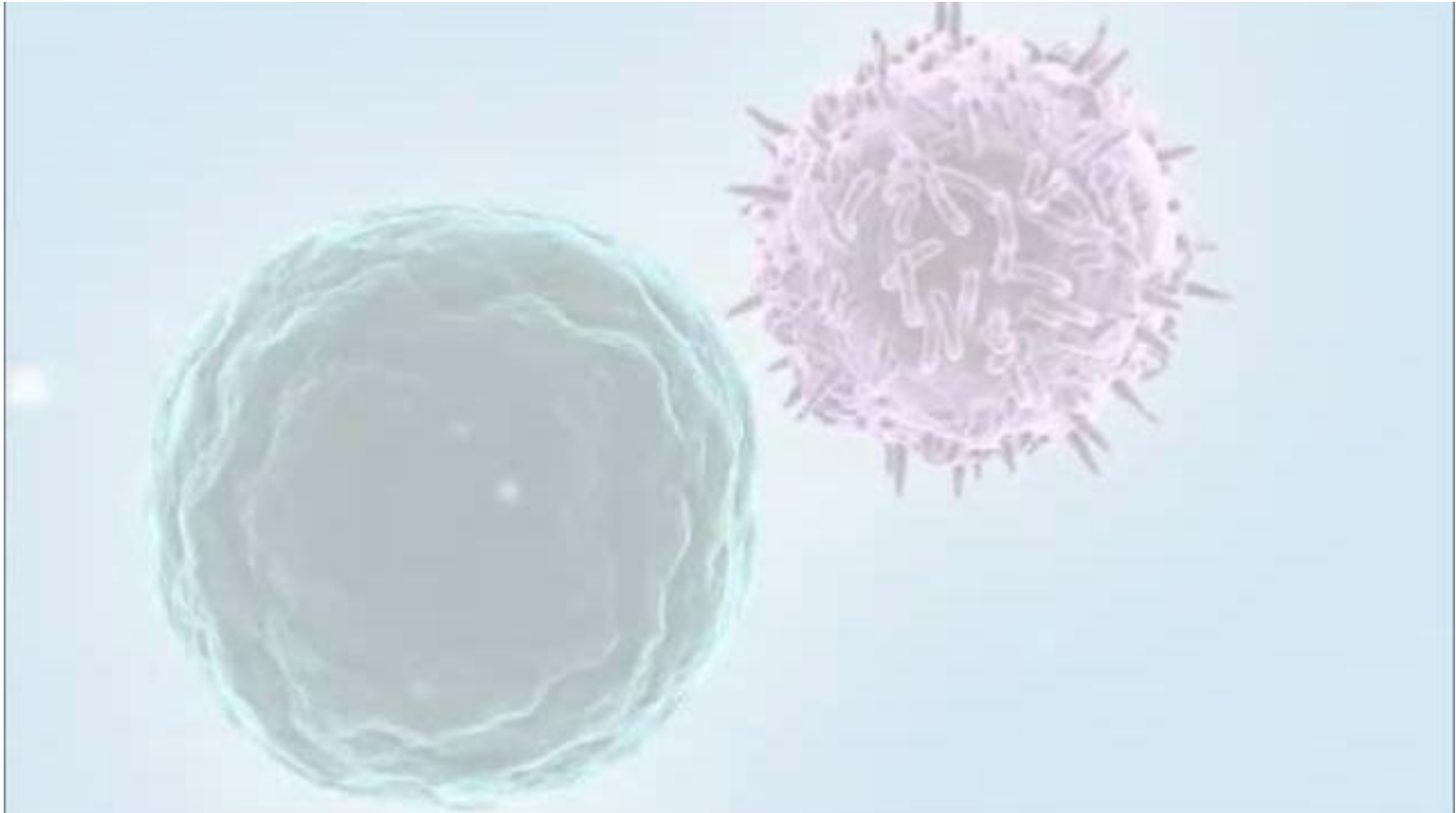
# Modelling Biophysics: Thresholding



# Modelling Biophysics: Thresholding



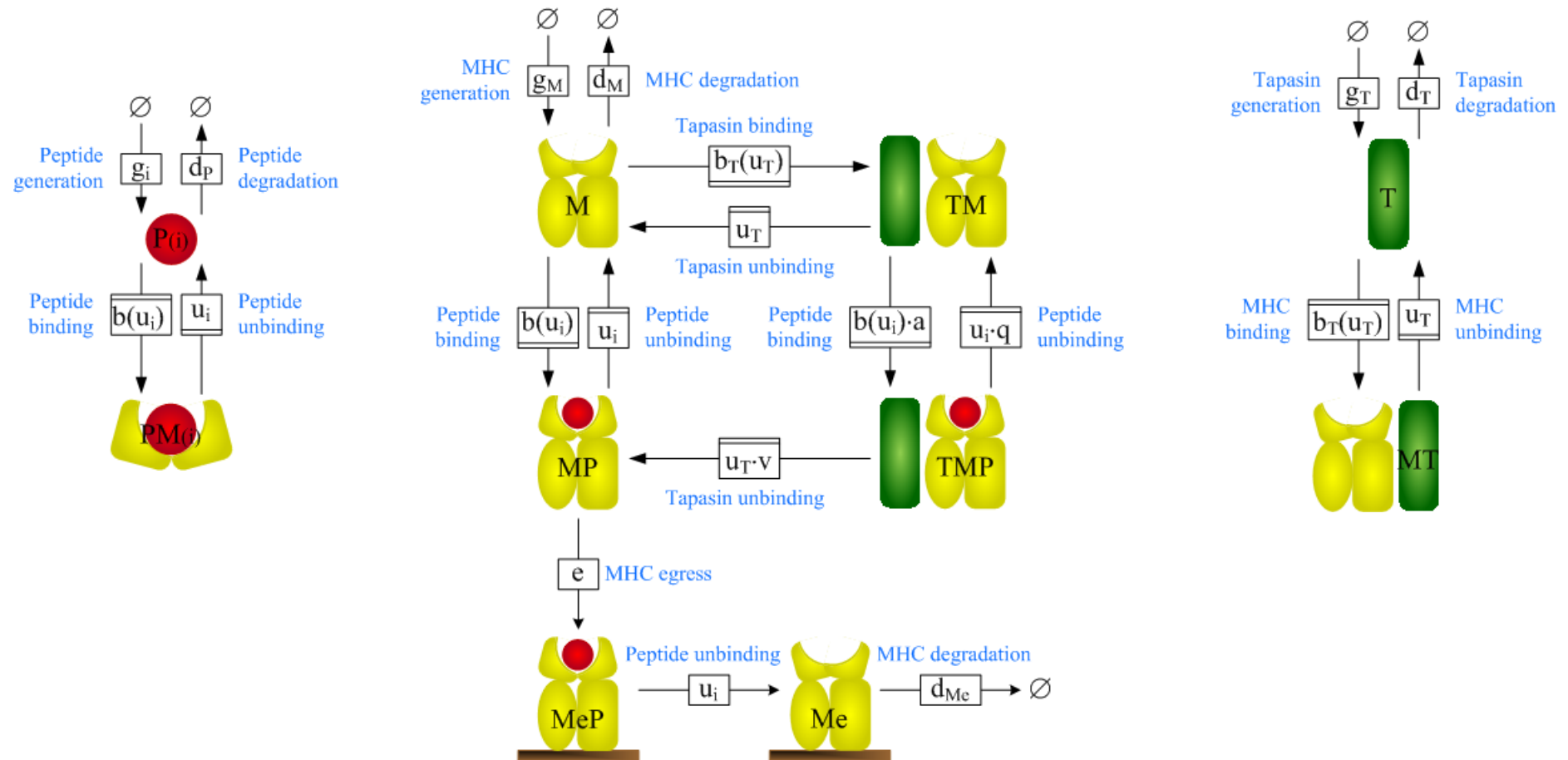
# MHC class I Antigen Presentation



# Peptide Optimisation

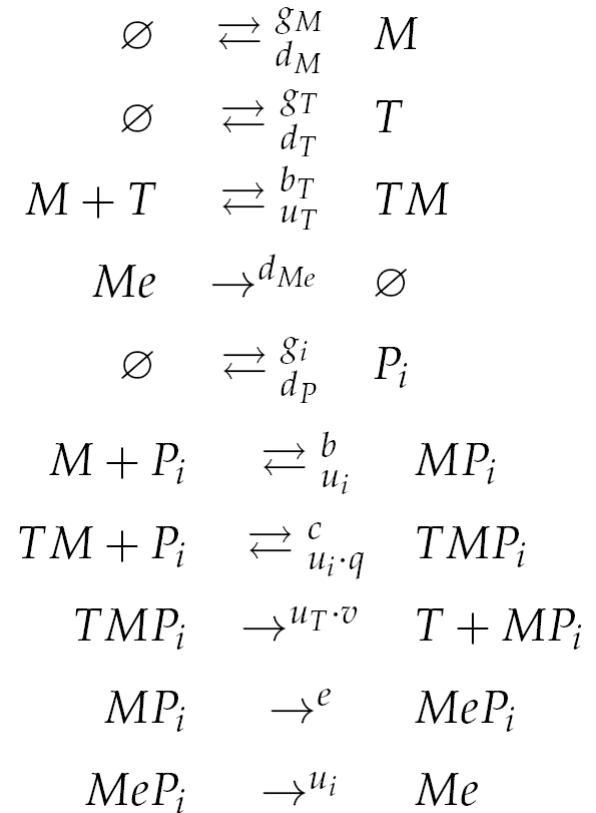
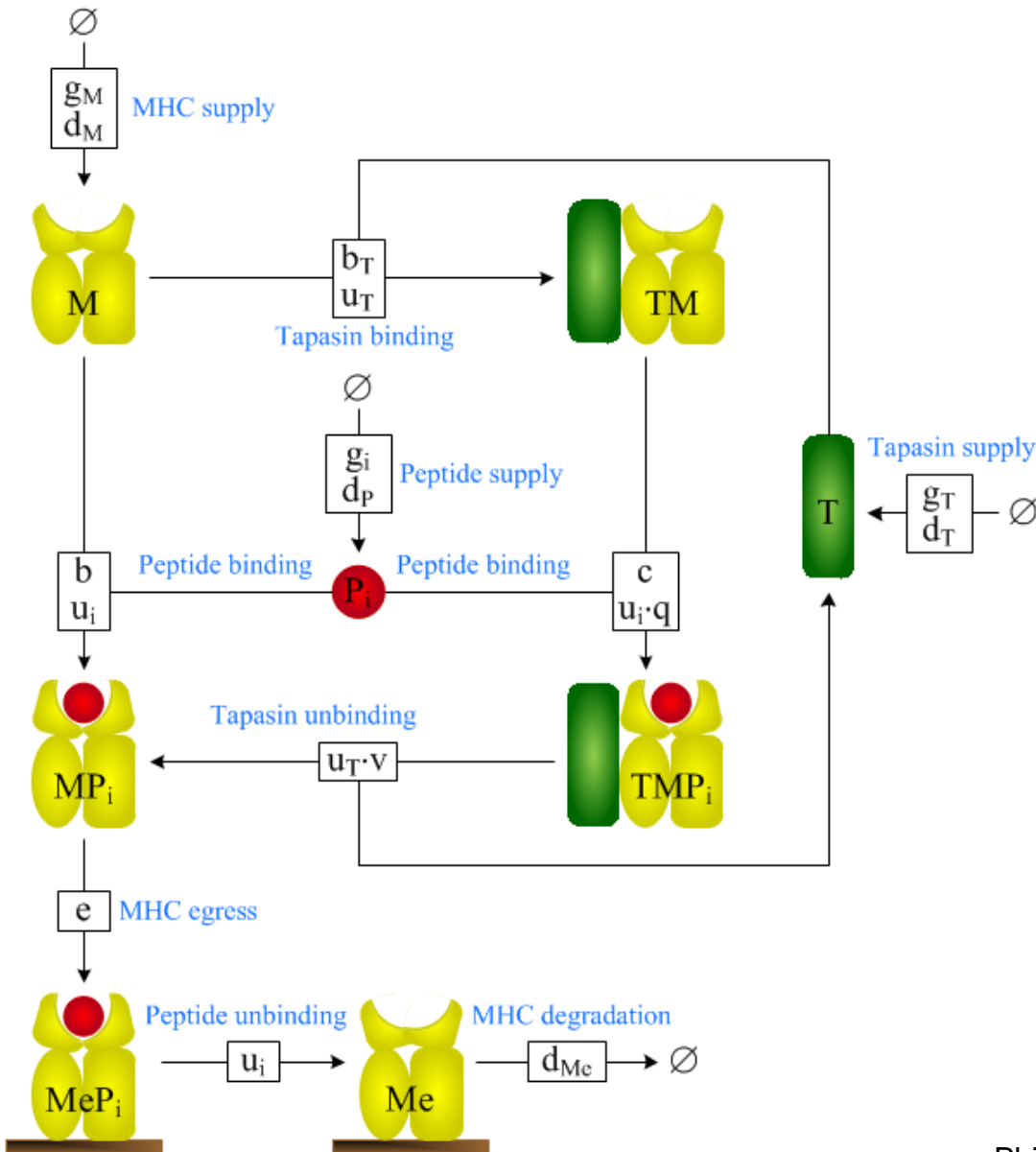
- Peptide-MHC complexes generally need to be stable for many hours or days at the cell surface
- *Peptide optimisation*: high affinity peptides are preferentially selected for presentation
- Optimisation in the ER is typically limited to tens of minutes
  - How is such high optimisation is achieved in so little time?
  - What are the main mechanisms of optimisation?

# MHC Class I Model

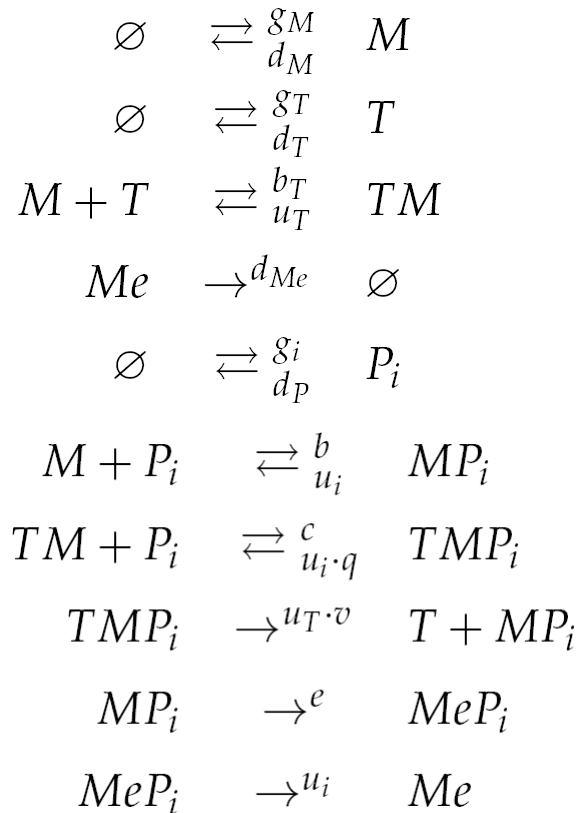




# Automatic generation of reactions



# Automatic generation of ODEs



$$[M]' = \sum_i u_i [MP_i] + u_T [TM] + g_M$$

$$- (b \sum_i [P_i] + b_T [T] + d_M) [M]$$

$$[T]' = u_T [TM] + g_T + u_T v \sum_i [TMP_i] - (b_T [M] + d_T) [T]$$

$$[MP_i]' = b [M] [P_i] + u_T v [TMP_i] - (u_i + e) [MP_i]$$

$$[TM]' = b_T [M] [T] + q \sum_i u_i [TMP_i] - (u_T + c \sum_i [P_i]) [TM]$$

$$[TMP_i]' = ba [TM] [P_i] - (u_i q + u_T v) [TMP_i]$$

$$[P_i]' = u_i [MP_i] + u_i q [TMP_i] + g_i$$

$$- (b [M] + c [TM] + d_p) [P_i]$$

$$[MeP_i]' = e [MP_i] - u_i [MeP_i]$$

$$[Me]' = \sum_i u_i [MeP_i] - d_{Me} [Me]$$

# Steady-state ODE analysis

$$[MeP_i]^* = \frac{1}{u_i} \frac{e}{u_i + e} \left( b[M]^* + \frac{x}{u_i + x} c[TM]^* \right) [P_i]^*$$

$$[TM]^* \gg [M]^* \quad \downarrow \quad [P_i]^* \approx g_i/d_P$$

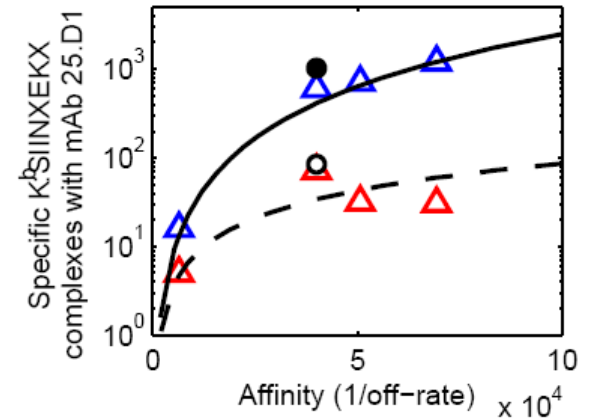
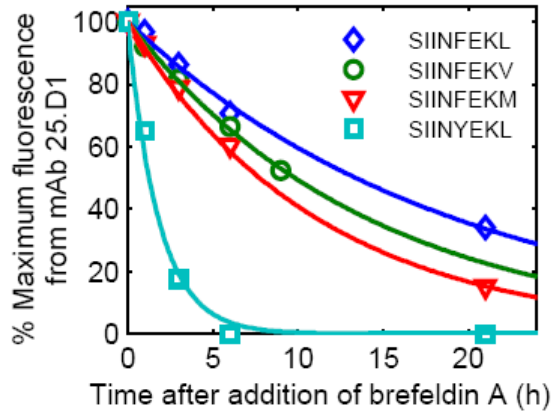
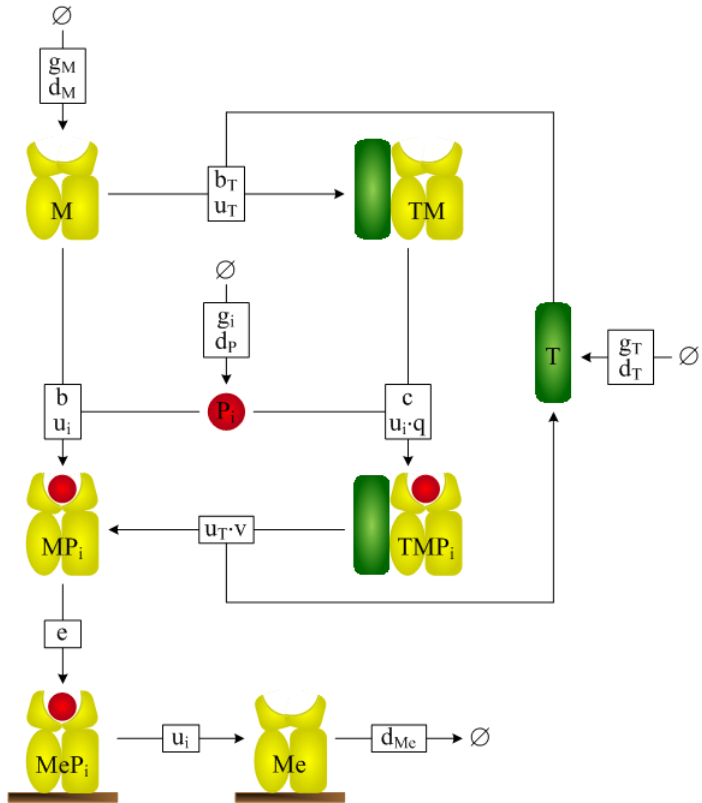
$$[MeP_i]^* \approx C \quad \text{Supply} \quad \frac{x}{u_i + x} \quad \text{Tapasin} \quad \frac{e}{u_i + e} \quad \text{ER} \quad \frac{1}{u_i} \quad \text{Surface}$$

$$x = u_T v / q \quad C = c[TM]^* / d_P$$

# Steady-state experiments

Tapasin enhances peptide presentation by  $1/u_i$

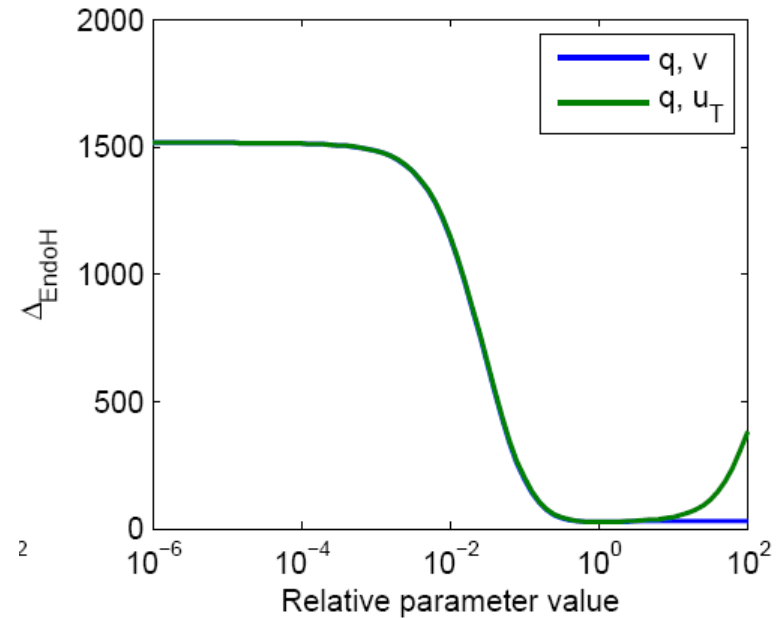
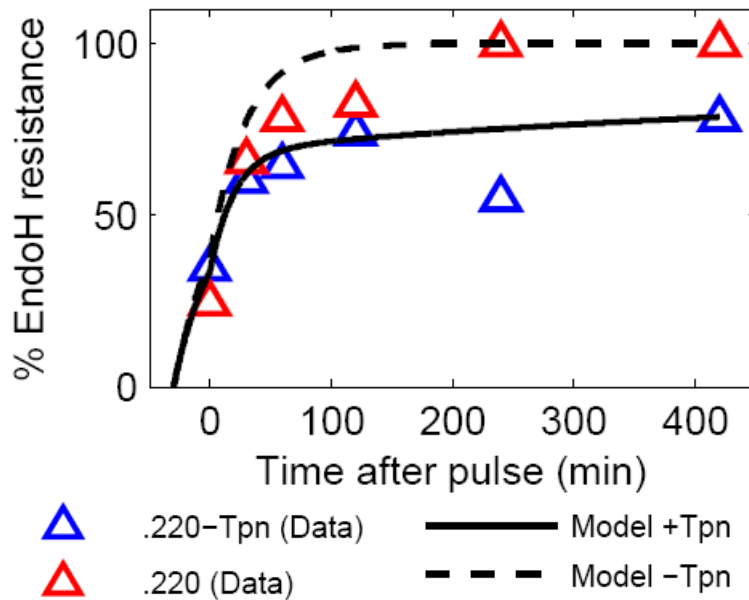
Measured off-rates  $u_i$  for SIINFEKL peptides



- ▲ .220-Tpn (Data)
- ▲ .220 (Data)
- Model +Tpn
- - - Model -Tpn
- Model +Tpn (2.5x peptide supply)
- Model -Tpn (2.5x peptide supply)

# Delayed egress vs. Enhanced off-rate

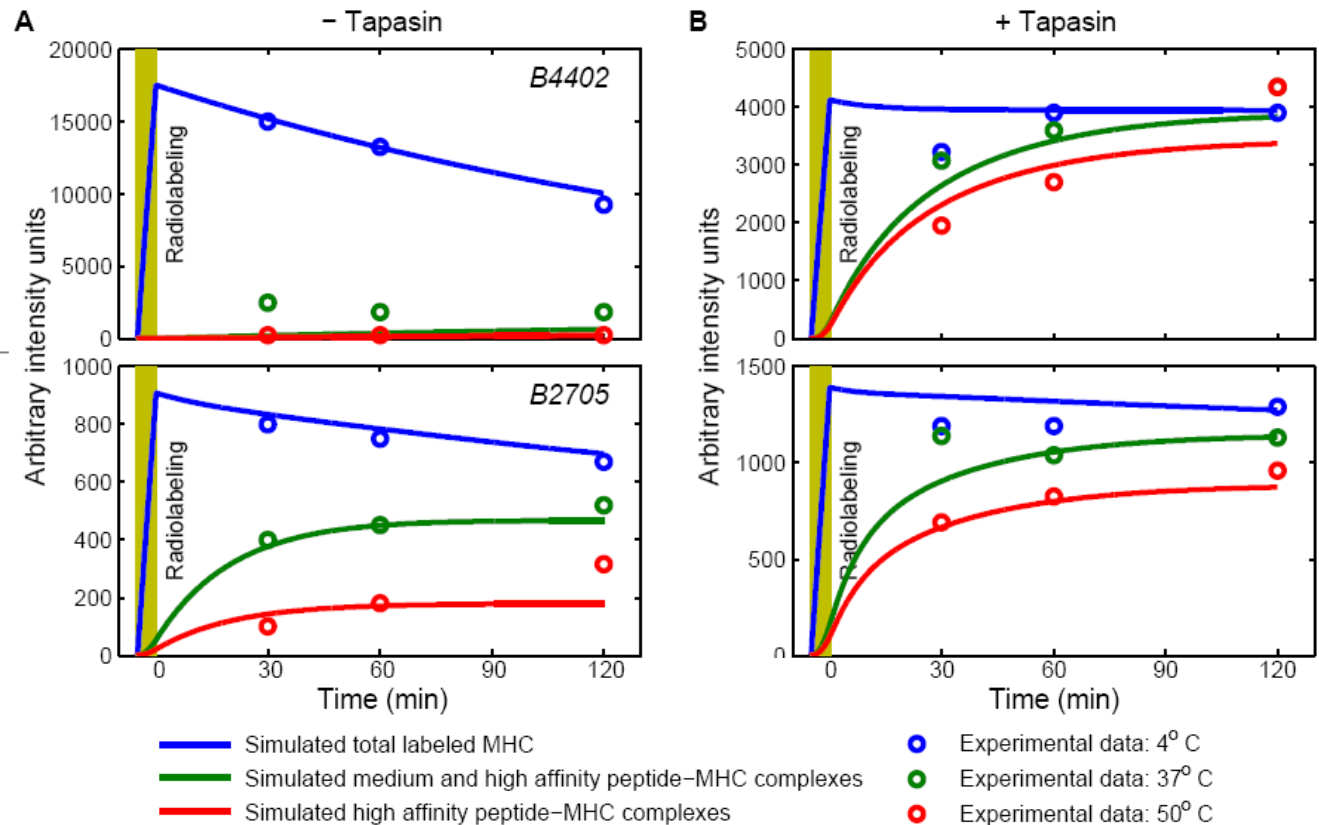
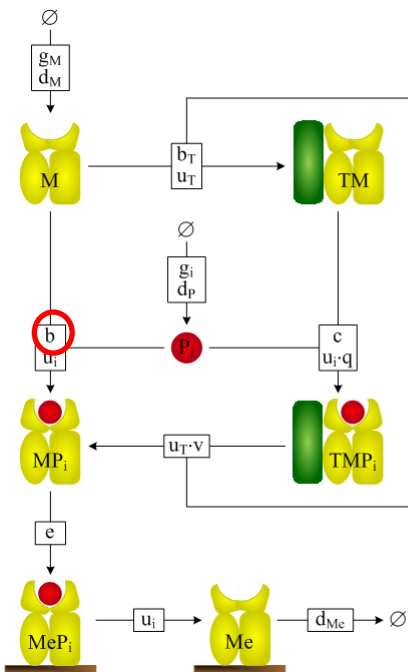
Measured transit time of MHC to cell surface



Fast transit => fast optimisation => enhanced off-rate

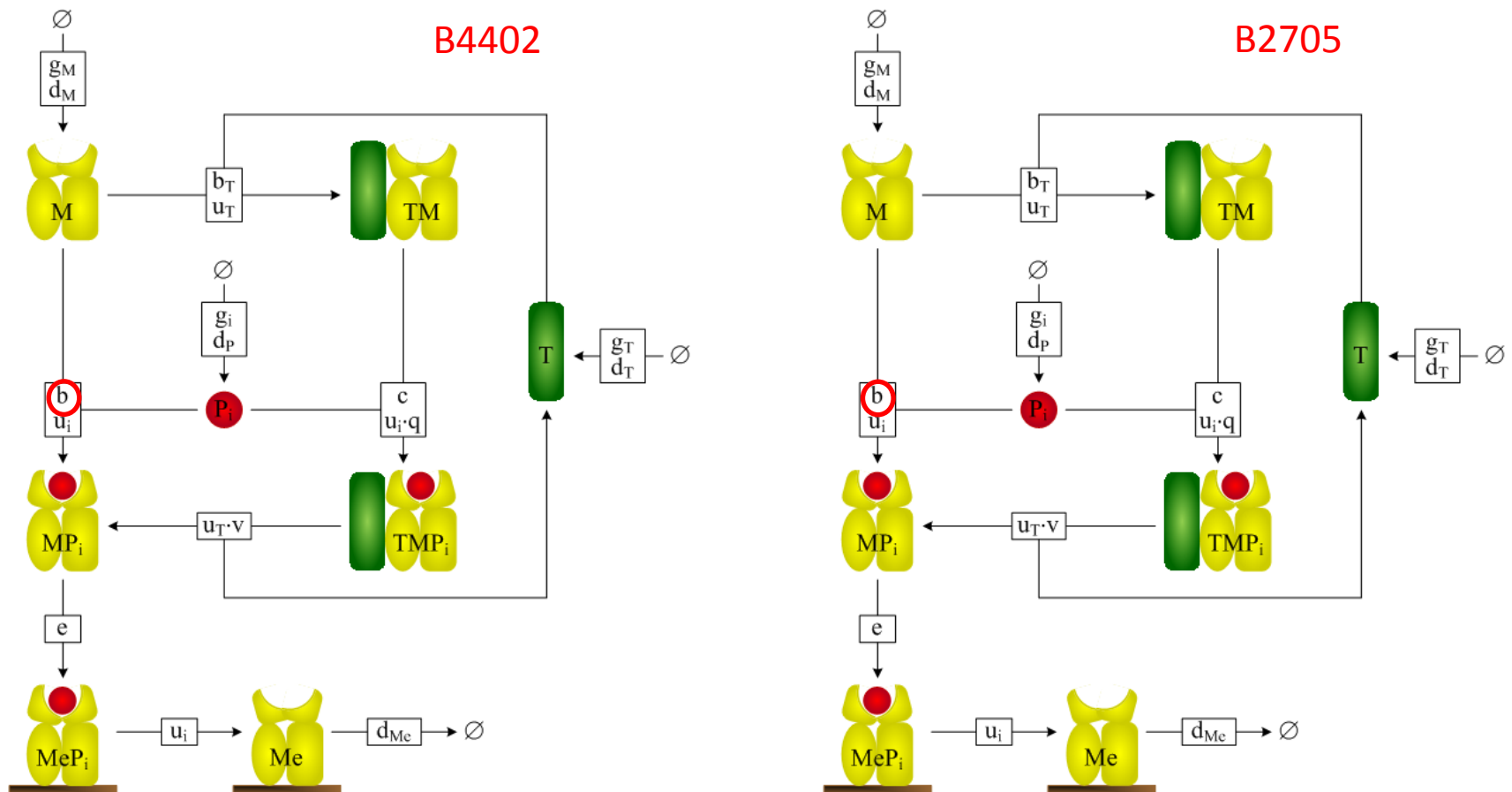
# Time-dependent experiments

Representative peptides  $P_{\text{low}}$ ,  $P_{\text{med}}$ ,  $P_{\text{high}}$



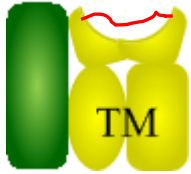
# Identify differences between alleles

## Allele-specific peptide binding





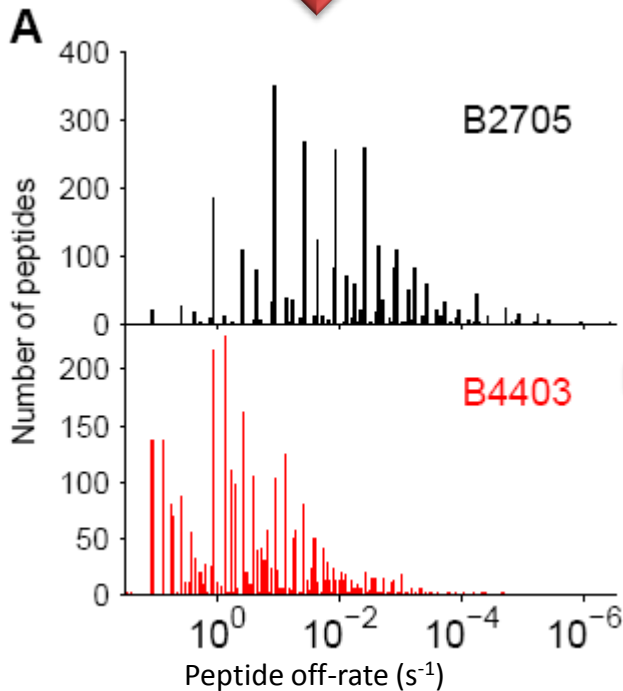
# Optimisation of HIV peptides



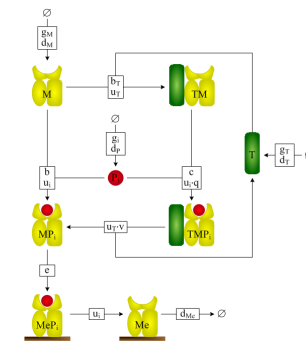
BIMAS Off-rate predictor

MGARASVLSGGELDRWEKIRLRPPGGKKKYK....

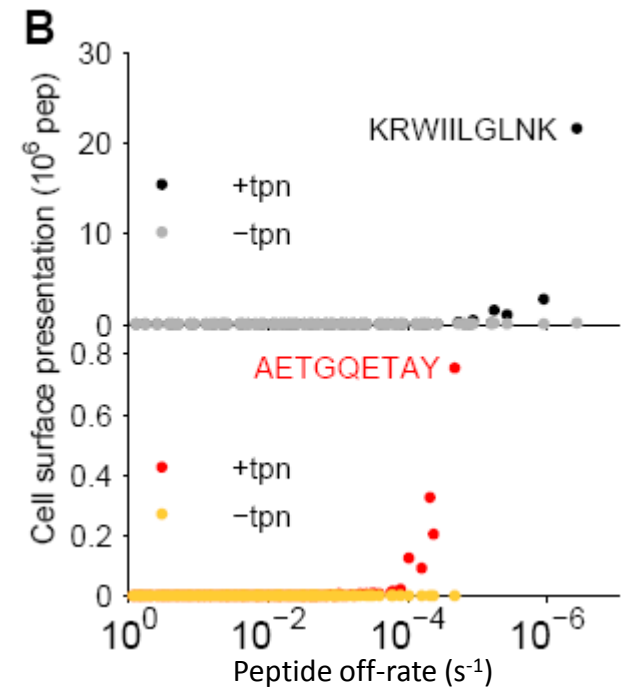
Viral protein sequence



Off-rates and abundance



Kinetic model



Cell-surface presentation

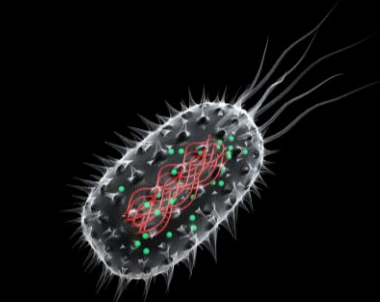
# Biological Computation Group

Molecule

Cell

Organ

Organism



DNA Computing

Synthetic Biology

Developmental Biology

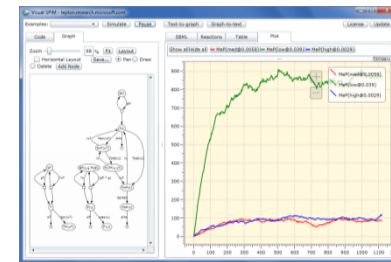
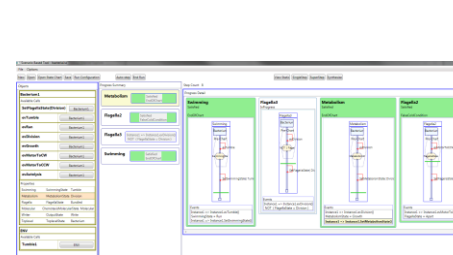
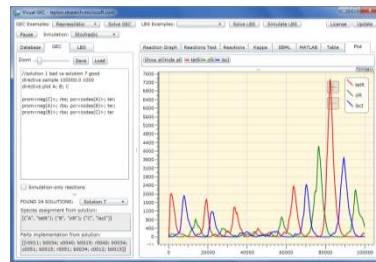
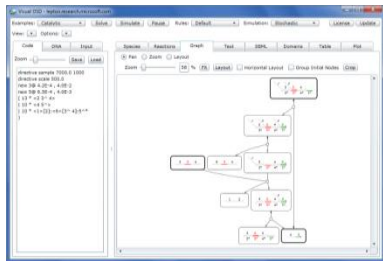
Immunology

**DSD: DNA Circuits**

**GEC: Genetic Devices**

**SBT: Stem Cells**

**SPiM: Processes**



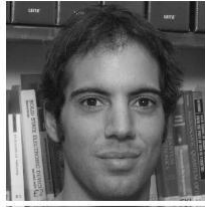
**Biological Modelling Engine: A common language runtime for Biological Computation**

# Acknowledgements

## DNA Computing



Luca  
Cardelli



Simon  
Youssef

## Synthetic Biology (Cambridge)



Michael  
Pedersen



James  
Brown



Tim  
Rudge

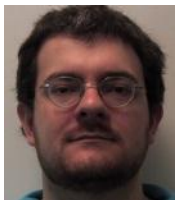


Jim  
Haseloff

## Microsoft Research



Matthew  
Lakin



Filippo  
Polo



Neil  
Dalchau



Stephen  
Emmott

## Immunology



Leonard  
Goldstein  
(Cambridge)



Mark  
Howarth  
(Oxford)



Tim  
Elliott  
(Southampton)



Joern  
Werner

