



# Algebras and Languages for Molecular Programming

Luca Cardelli  
Microsoft Research

Torino, 2010-03-24  
<http://lucacardelli.name>



# Smaller and Smaller

Dec. 23, 1947. John Bardeen and Walter Brattain show the first working transistor.

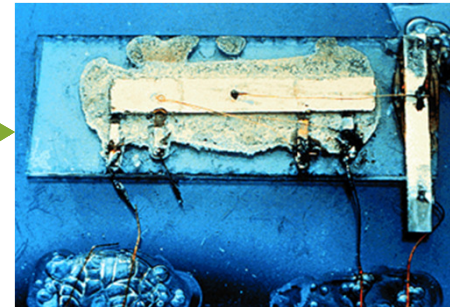
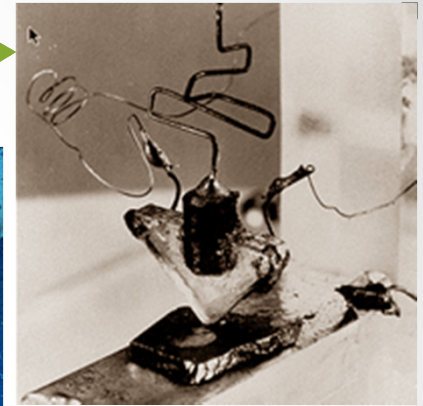
Sep. 1958. Jack Kilby builds the first integrated circuit.

50 years later

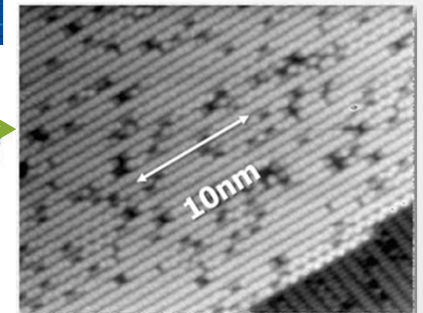
Jan. 2010. Intel and Micron announce 25nm NAND flash.

Dec. 24, 2009. Working transistor made of a single molecule.

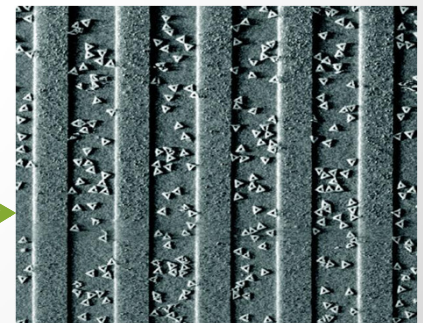
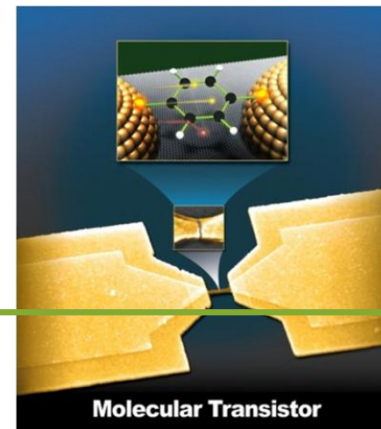
**<10 iterations of Moore's Law left!**  
The race is on for *molecular scale integrated circuits*.



Scanning tunneling microscope image of a silicon surface showing 10nm is ~20 atoms across



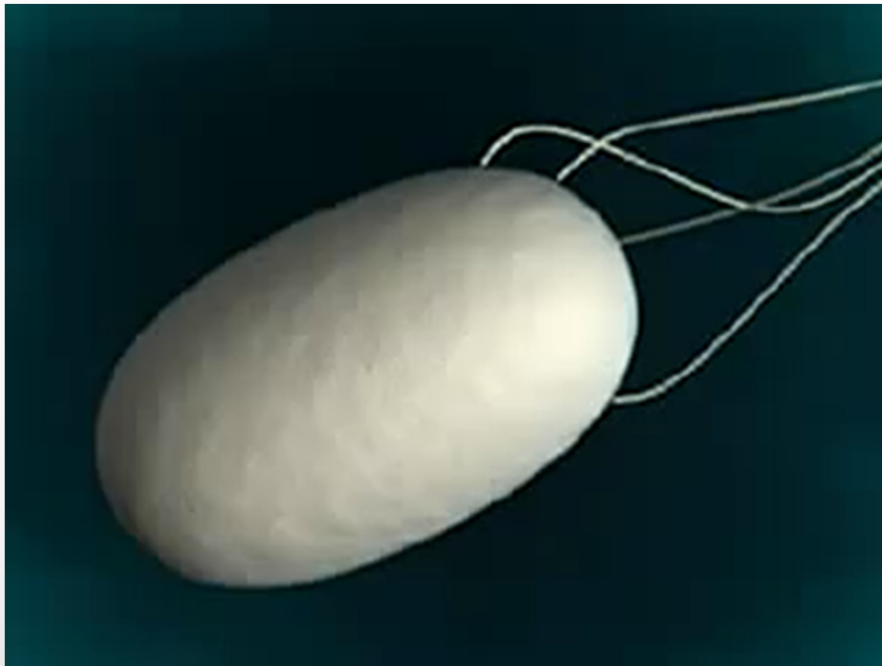
Observation of molecular orbital gating. *Nature*, 2009; 462 (7276): 1039



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

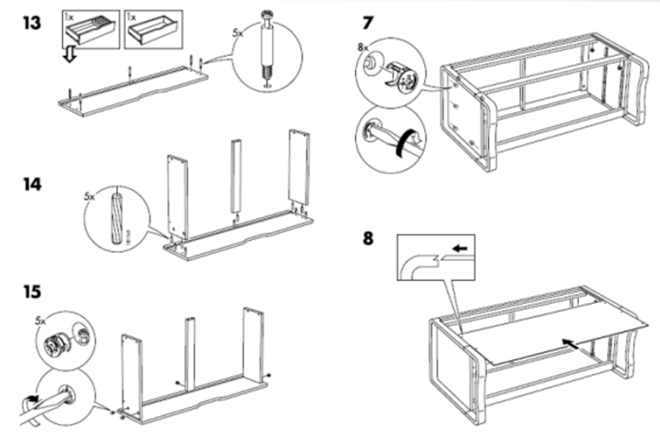
# Building The *Smallest* Things

- How do we build structures that are by definition smaller than your tools?
- Basic answer: you can't. Structures (and tools) should build themselves!
- By *programmed self-assembly*.

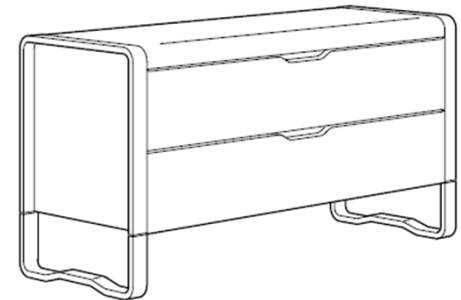


# Molecular IKEA

- Nature can self-assemble.  
Can we?
- *“Dear IKEA, please send me a chest of drawers that assembles itself.”*
- We need a magical material where the pieces are pre-programmed to fit into to each other.
- At the molecular scale many such materials exist; let’s pick one...

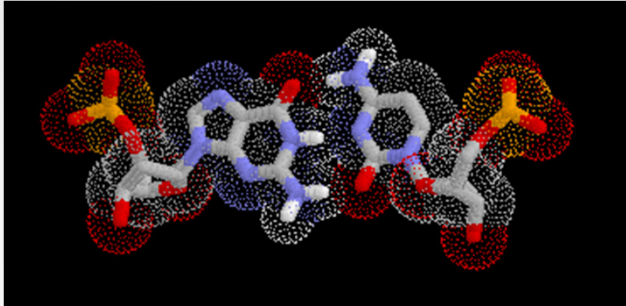


Add water

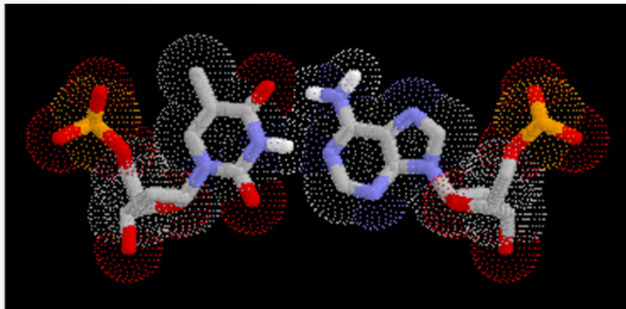




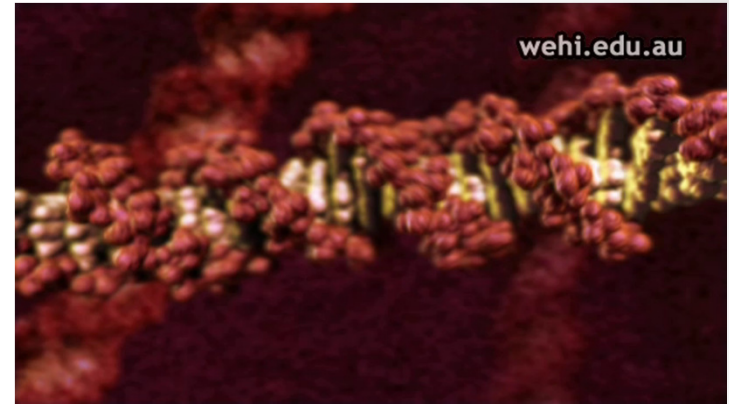
# DNA



GC Base Pair  
Guanine-Cytosine

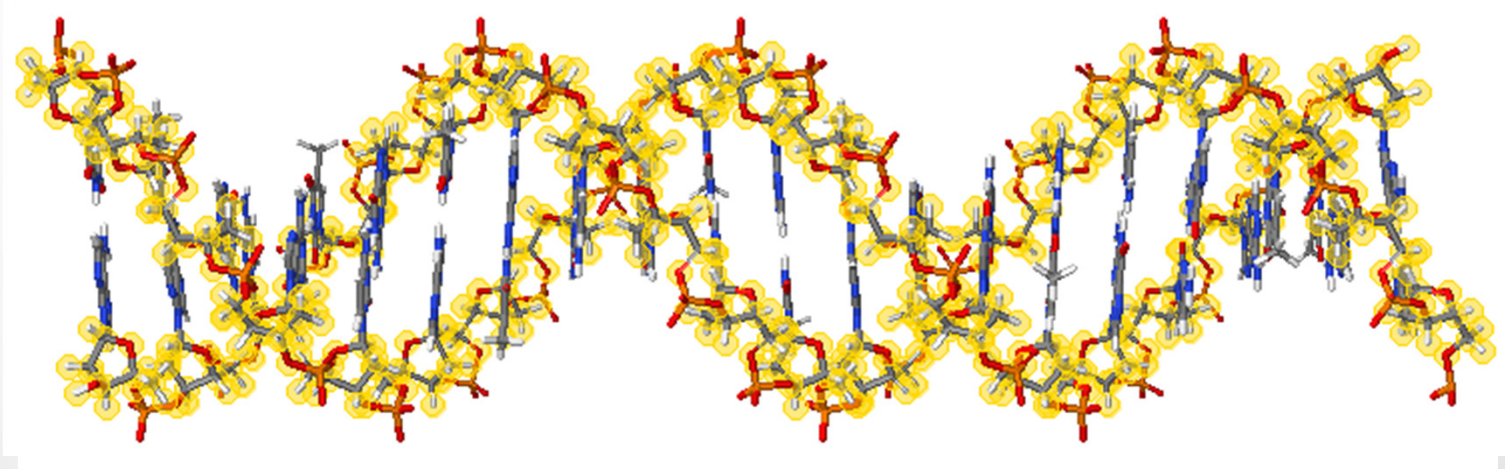


TA Base Pair  
Thymine-Adenine



[Interactive DNA Tutorial](http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html)

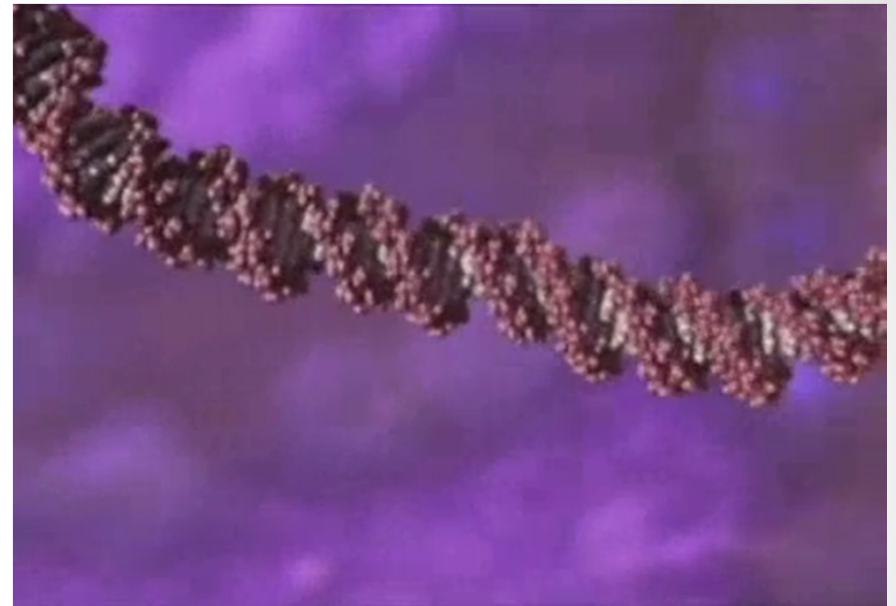
(<http://www.biosciences.bham.ac.uk/labs/minchin/tutorials/dna.html>)



Sequence of Base Pairs (GACT alphabet)

# Robust, and *Long*

- DNA in each human cell:
  - 3 billion base pairs
  - **2 meters long**, 2nm thick
  - folded into a 6 $\mu$ m ball
  - 750 MegaBytes
- A huge amount for a cell
  - Every time a cell replicates it has to copy *2 meters of DNA* reliably.
  - To get a feeling for the scale disparity, compute:
- DNA in human body
  - 10 trillion cells
  - 133 Astronomical Units long
  - 7.5 OctaBytes
- DNA in human population
  - 20 million light years long



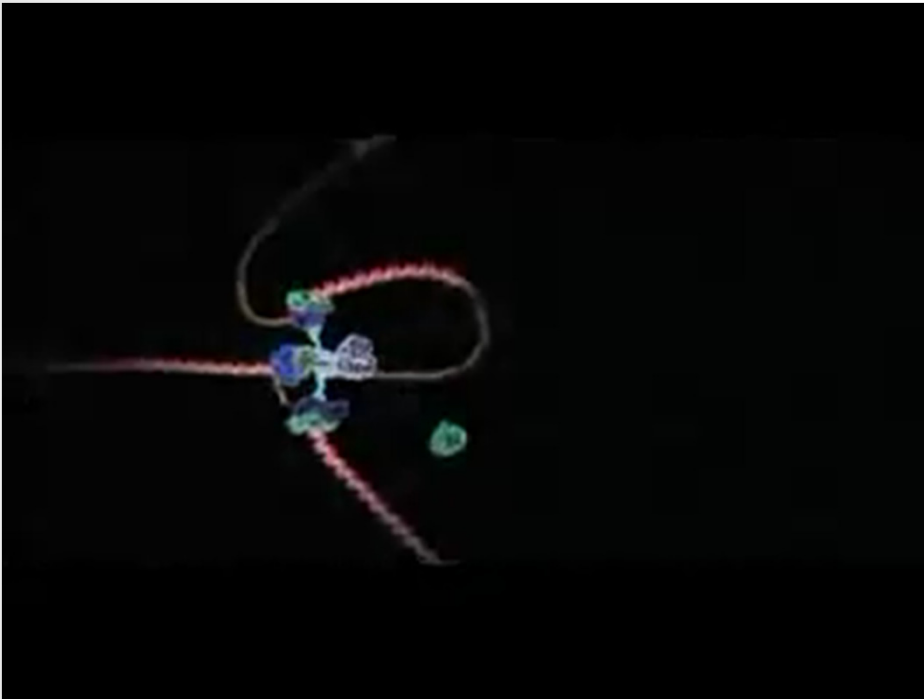
DNA wrapping into chromosomes



Andromeda Galaxy  
2.5 million light years

# Zippering Along

- DNA can support structural and computational complexity.



## DNA replication in *real time*

In Humans: 50 nucleotides/second  
Whole genome in a few hours (with parallel processing)

In Bacteria: 1000 nucleotides/second  
(higher error rate)



## DNA transcription in *real time*

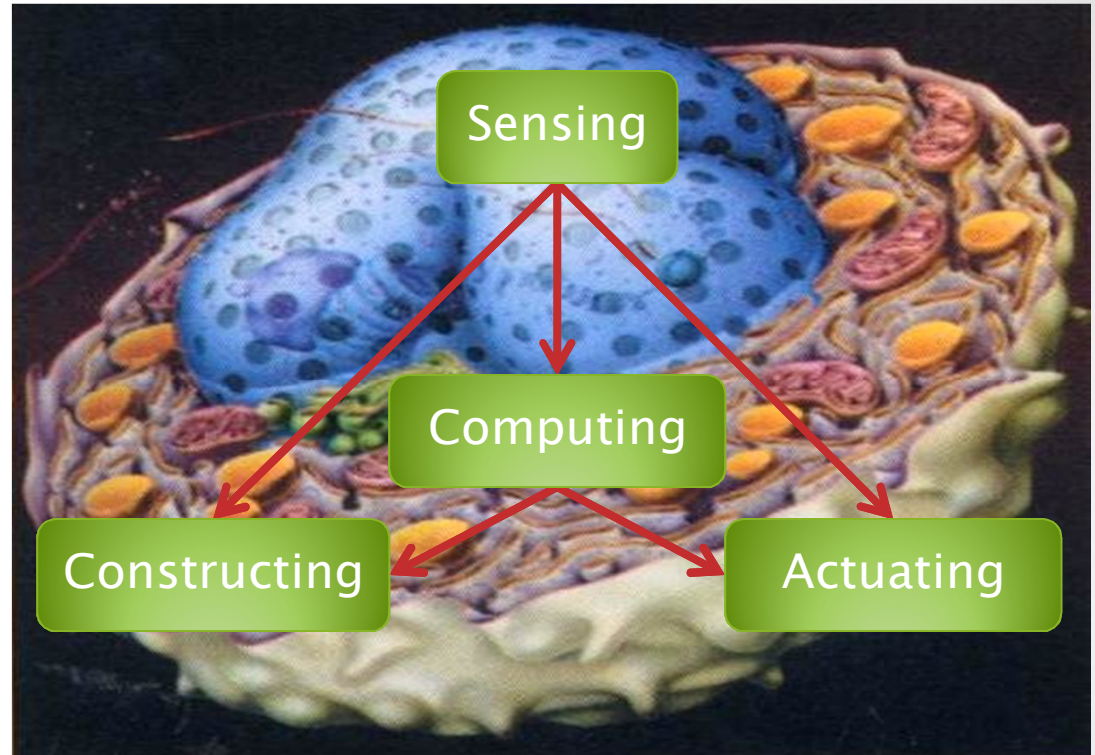
RNA polymerase II: 15–30 base/second

Drew Berry

<http://www.wehi.edu.au/wehi-tv>

# Nanoscale Engineering

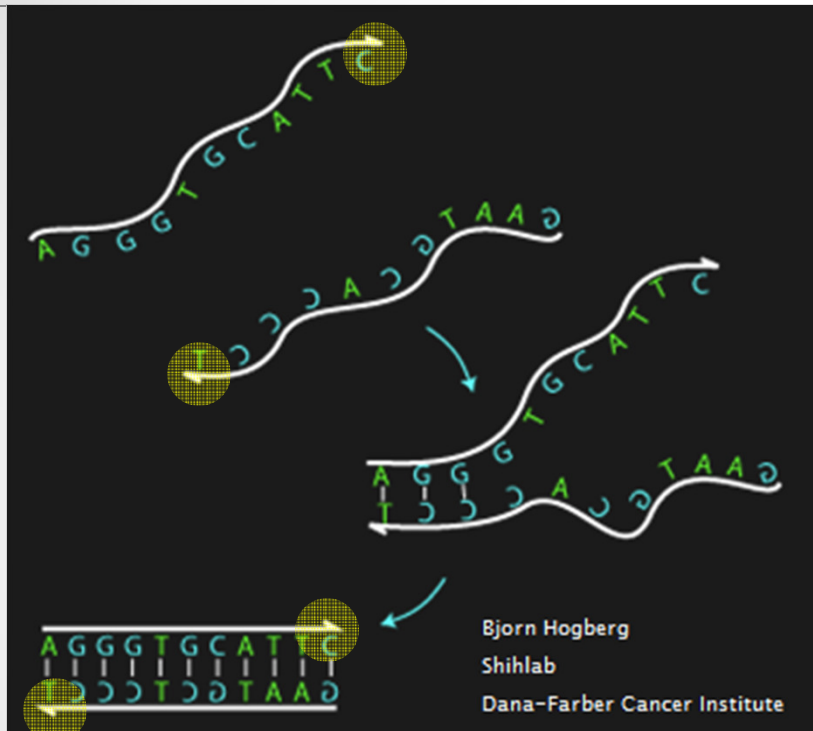
- Sensing
  - Reacting to forces
  - Binding to molecules
- Actuating
  - Releasing molecules
  - Producing forces
- Constructing
  - Chassis
  - Growth
- Computing
  - Signal Processing
  - Decision Making



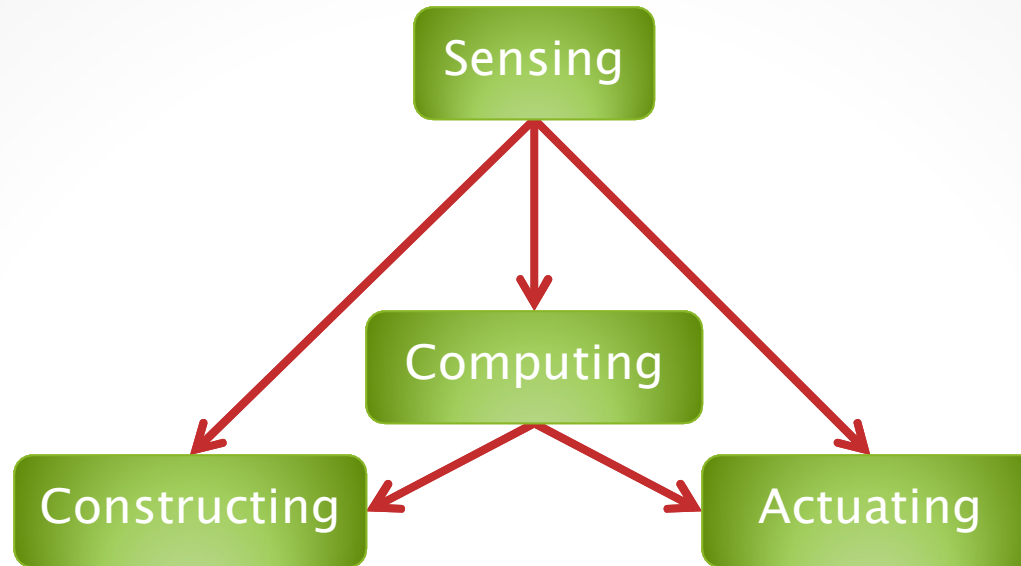
Nucleic Acids can do all this.  
And interface to **biology**.



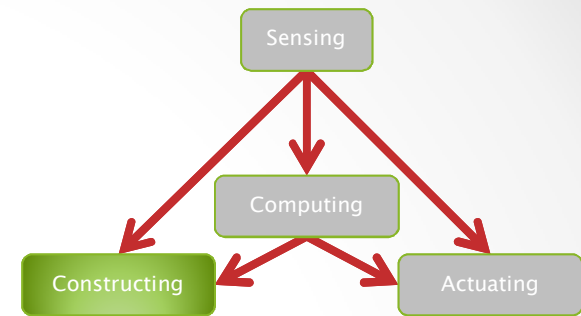
# Hybridization



- Strands with opposite orientation and complementary base pairs stick to each other (Watson-Crick duality).
- This is all we are going to use
  - We are not going to exploit DNA replication, transcription, translation, restriction and ligation enzymes, etc., which enable other classes of tricks.



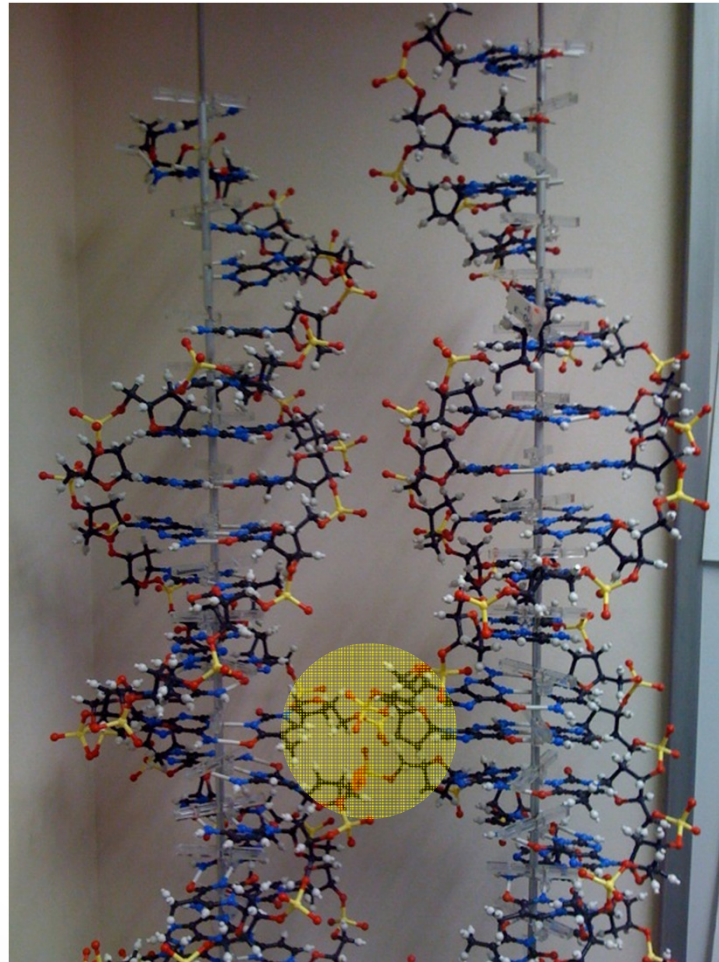
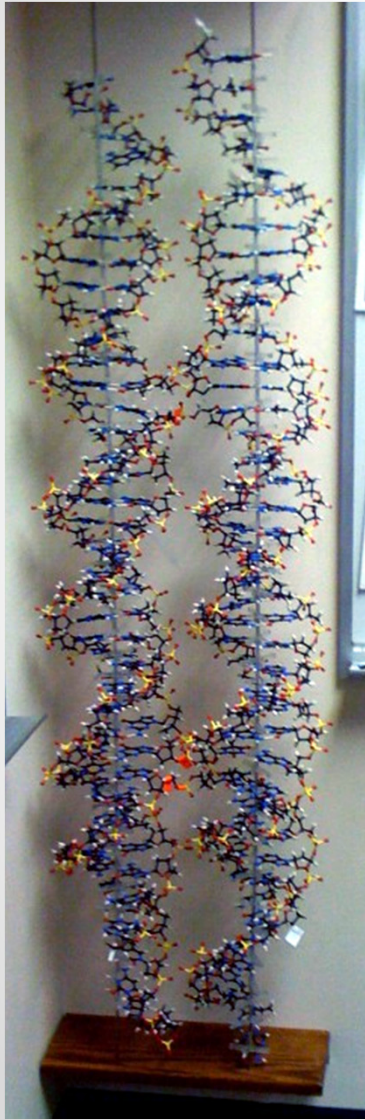
# Hybridization Tricks



# Constructing

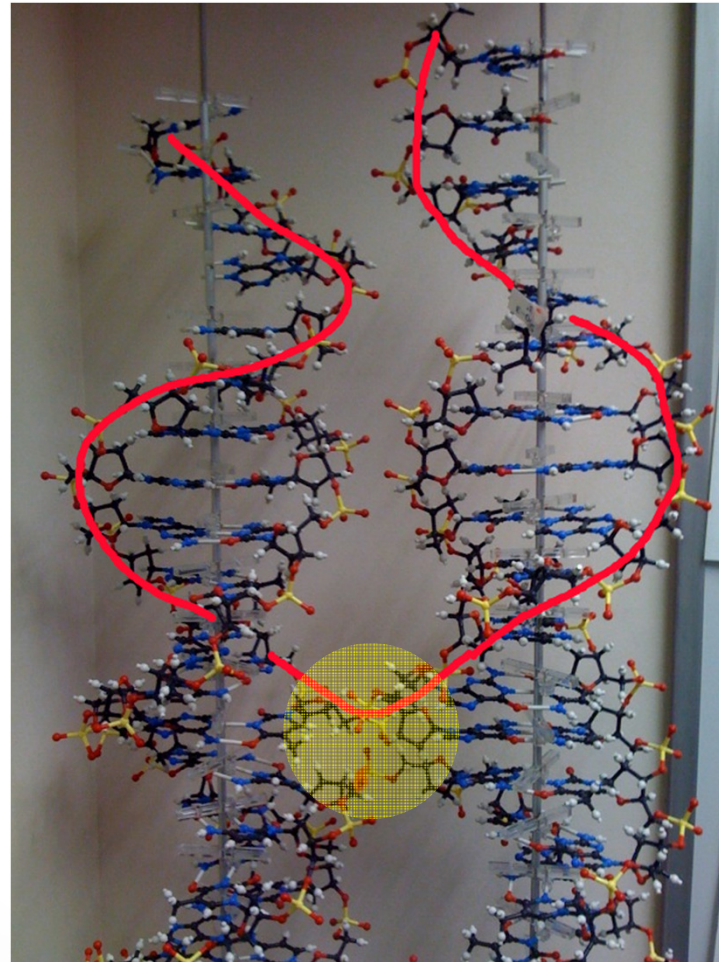
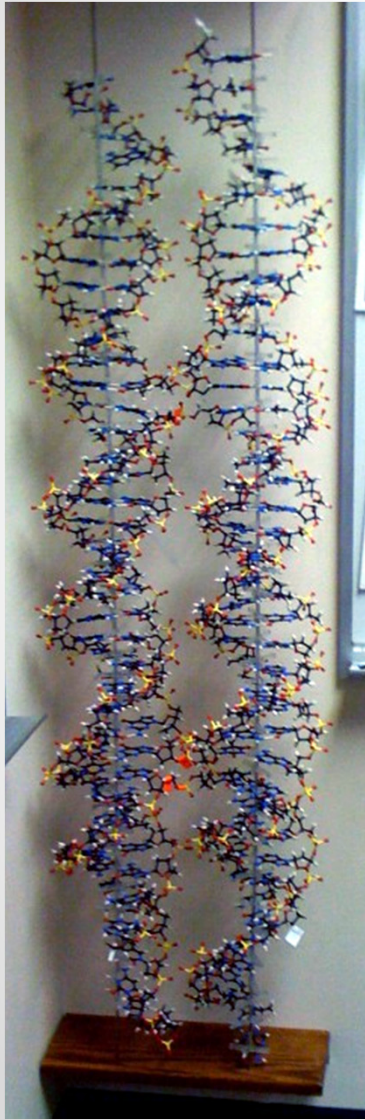
...

# Crosslinking

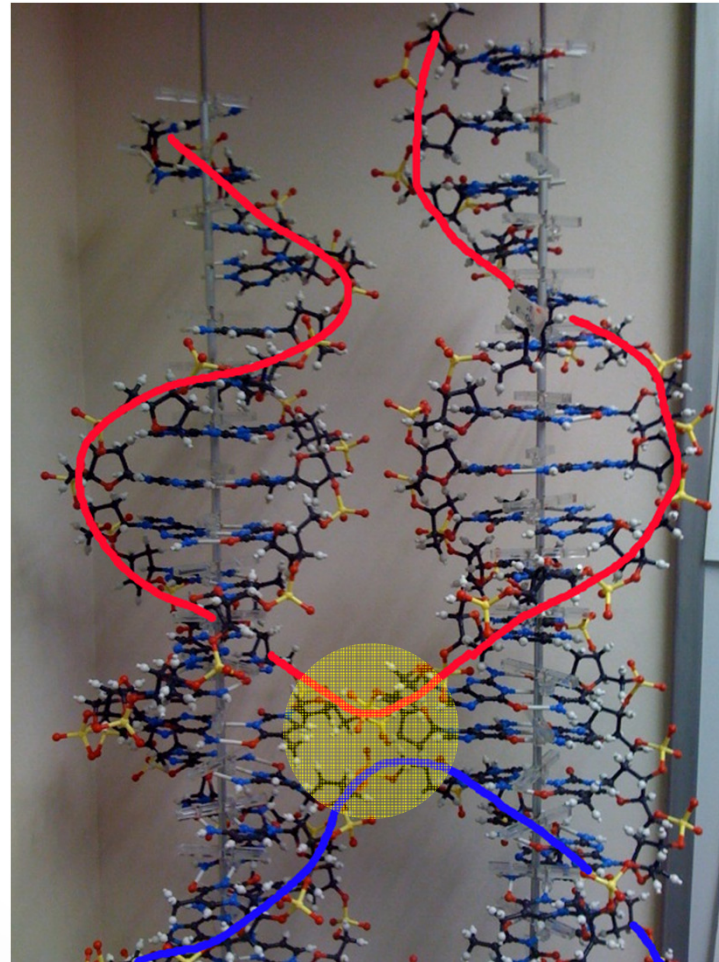




# Crosslinking

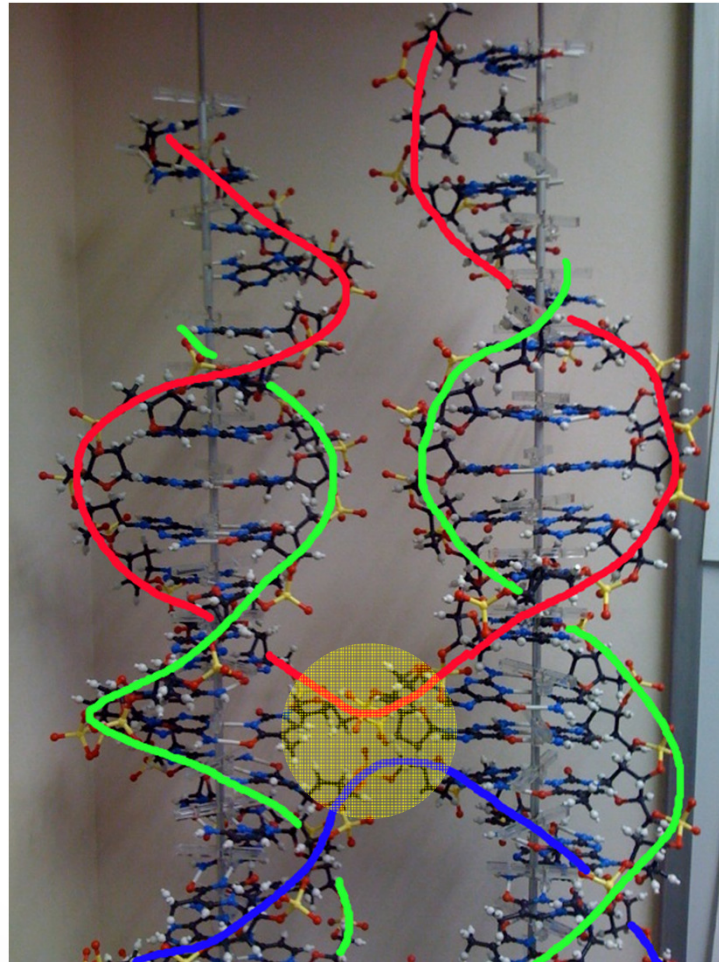
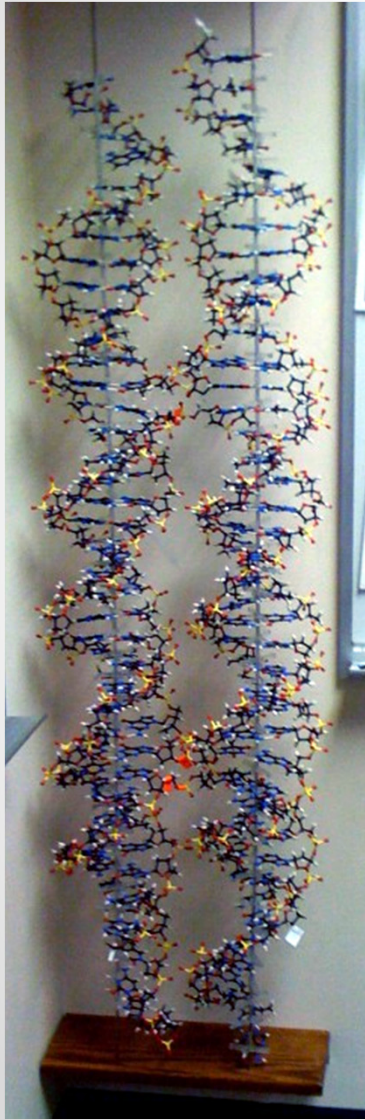


# Crosslinking

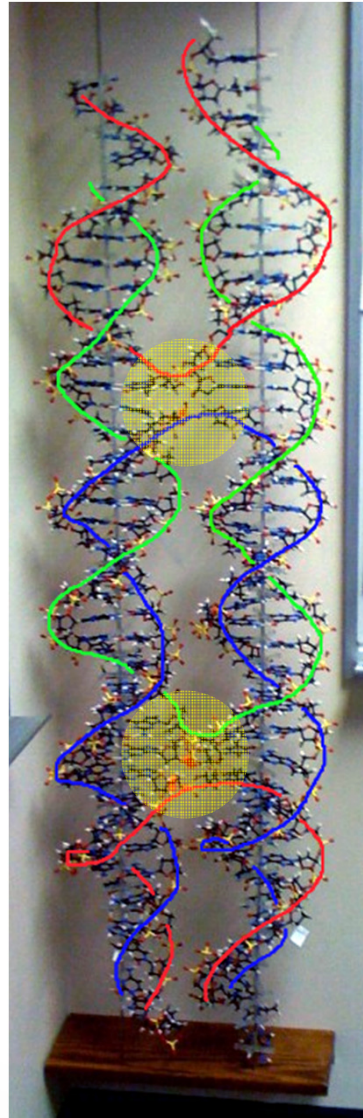
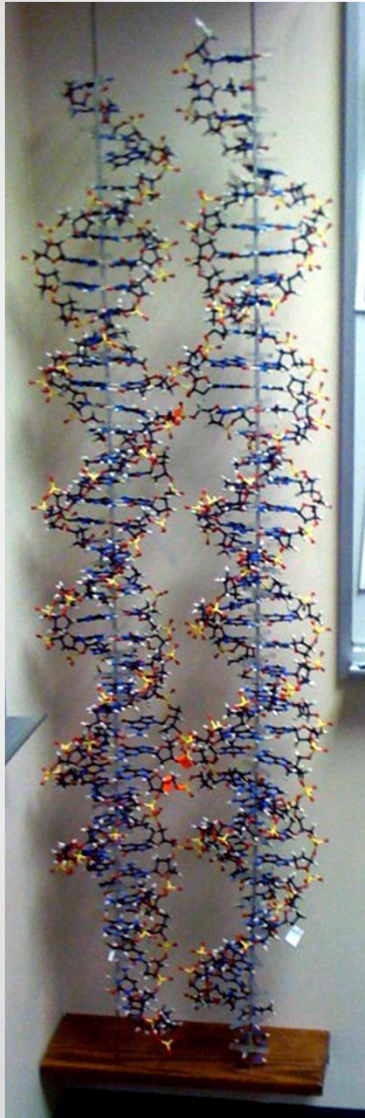




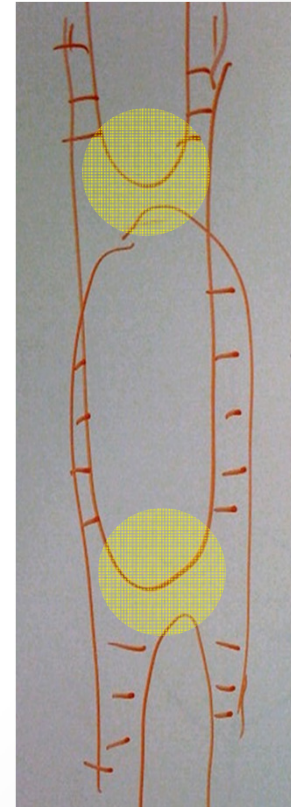
# Crosslinking



# Crosslinking



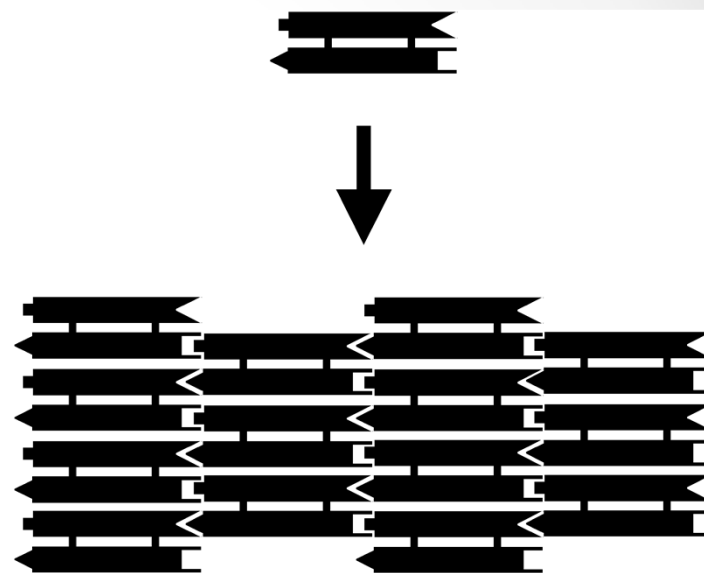
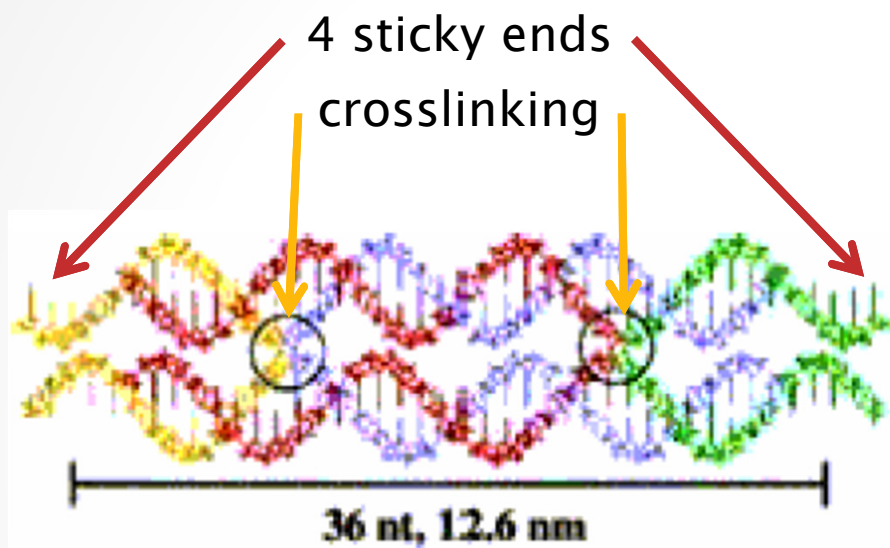
In nature, crosslinking is deadly (blocks DNA replication).



In engineering, crosslinking is the key to using DNA as a construction material.

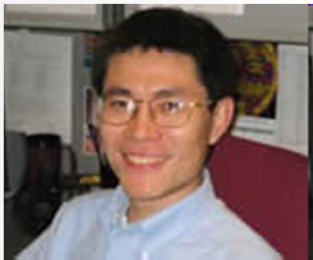


# DNA Tiling

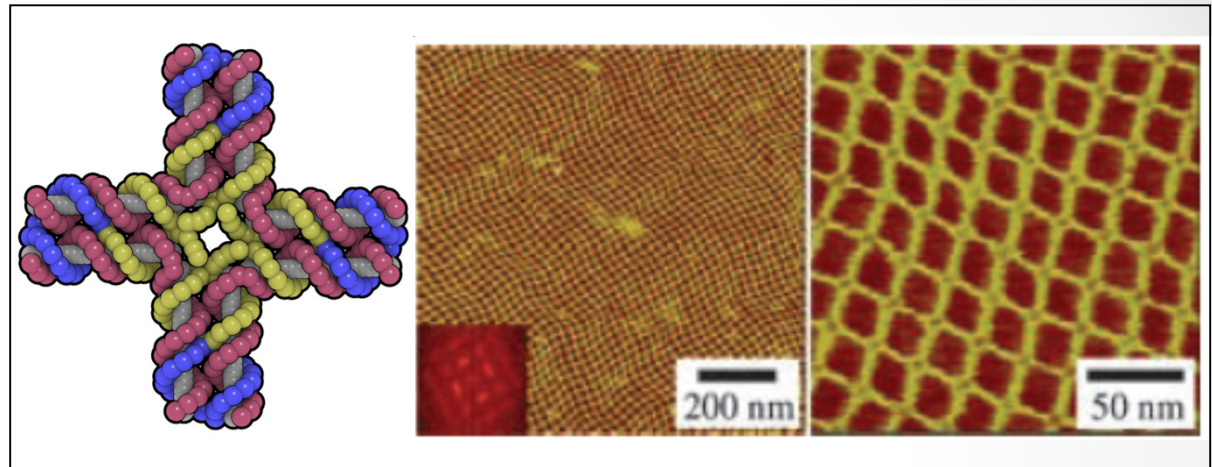


Construction and manipulation of DNA tiles in free space

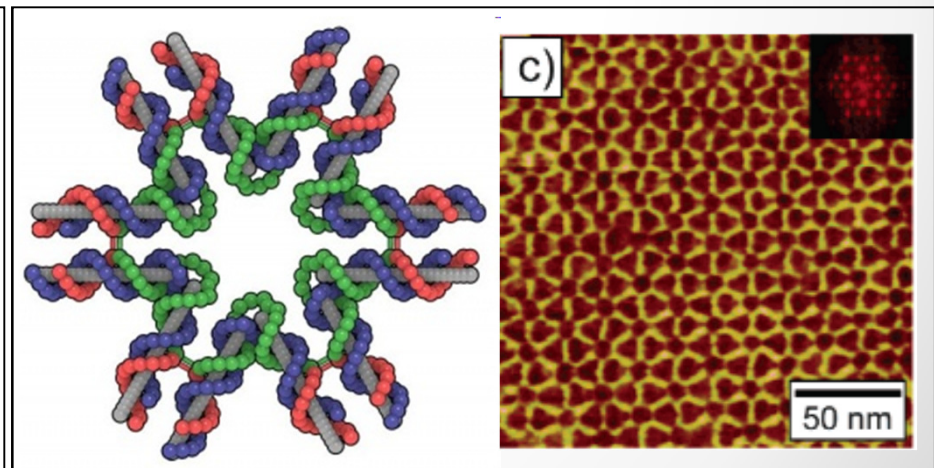
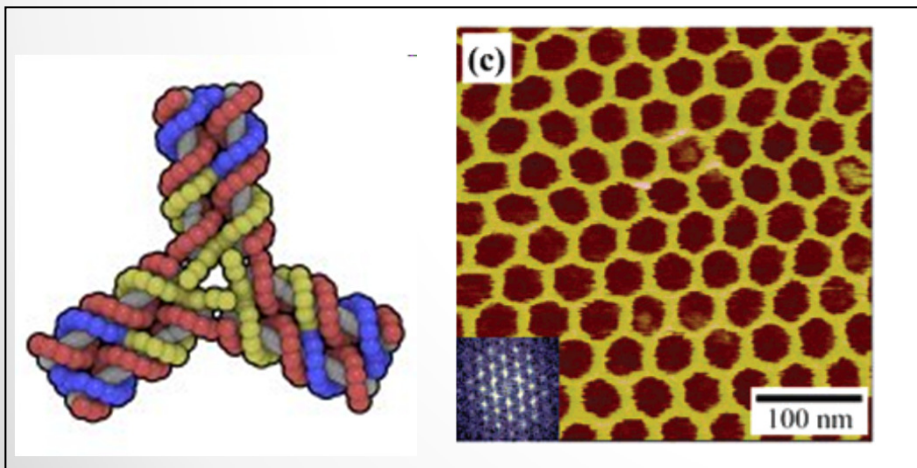
# 2D DNA Lattices



Chengde Mao  
Purdue University, USA



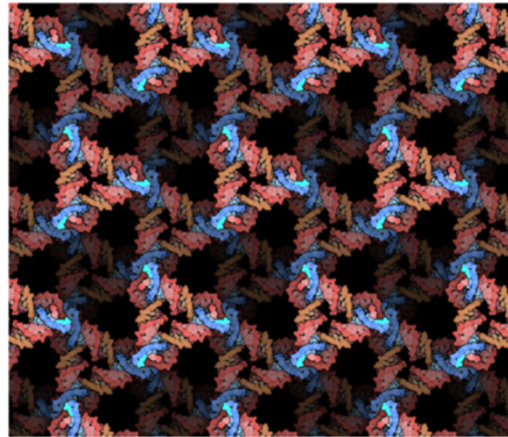
N-point Stars



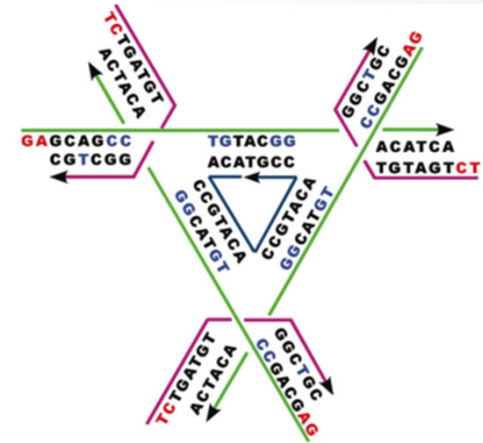
# 3D DNA Structures



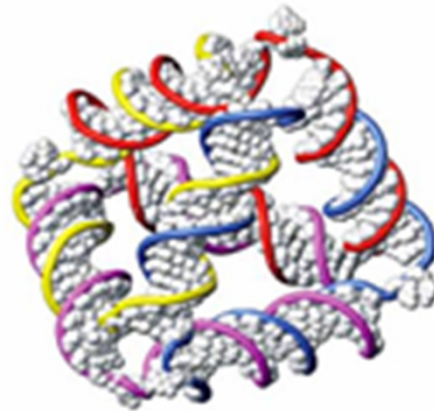
Ned Seeman  
NYU



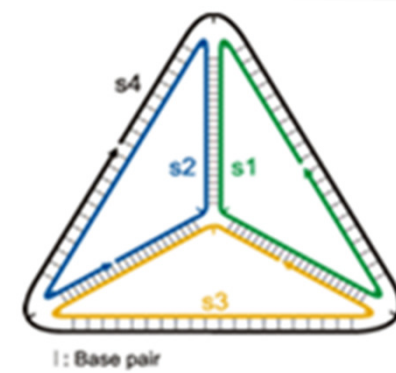
3D Crystal



Andrew Tuberfield  
Oxford



Tetrahedron

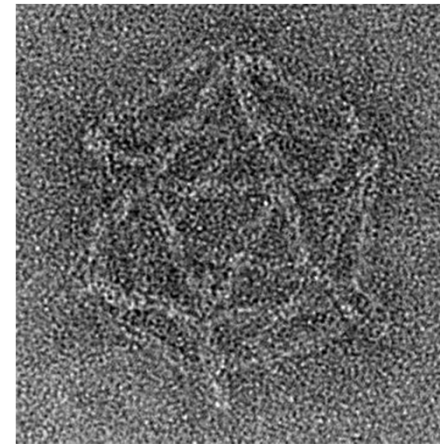
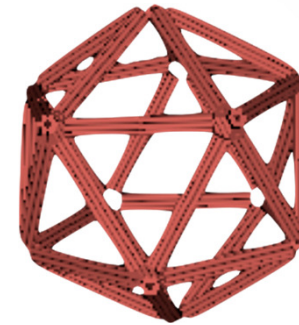
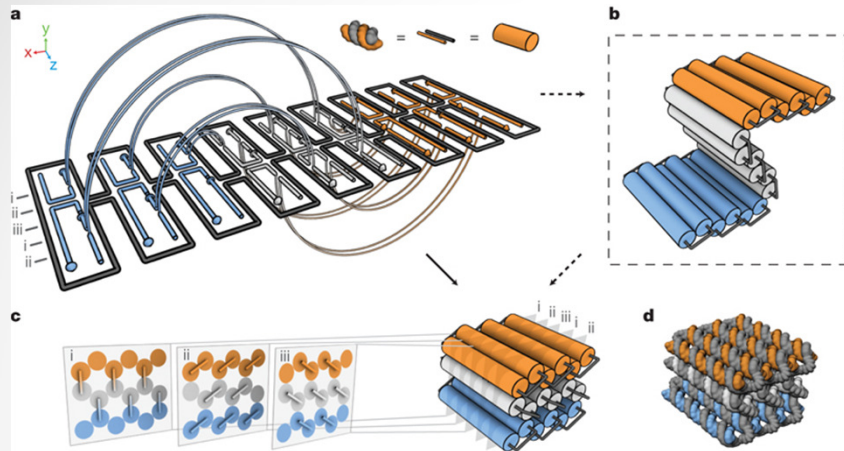




# CADnano



William Shih  
Harvard



## Folding DNA into Twisted and Curved Nanoscale Shapes

Hendrik Dietz, Shawn M. Douglas, & William M. Shih  
[Science, 325:725–730, 7 August 2009.](#)

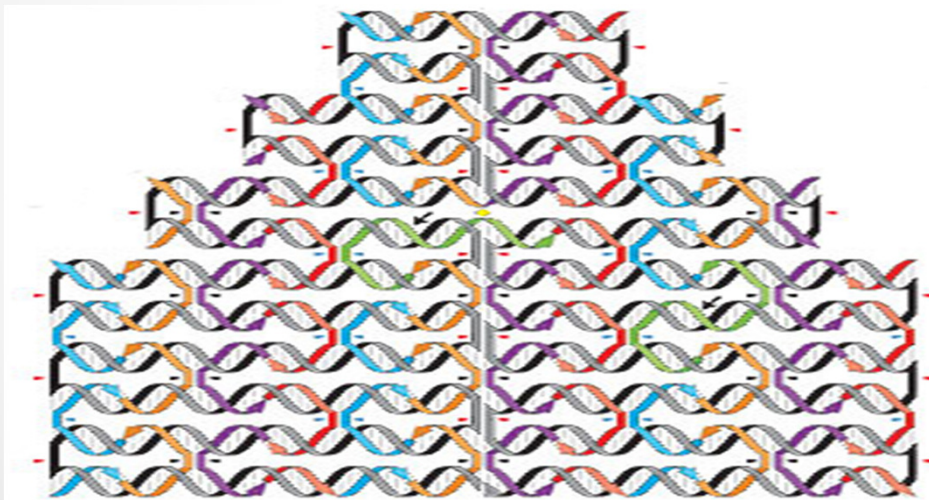


S.M. Douglas, H. Dietz, T. Liedl, B. Högberg, F. Graf and W. M. Shih  
Self-assembly of DNA into nanoscale three-dimensional shapes, *Nature* (2009)



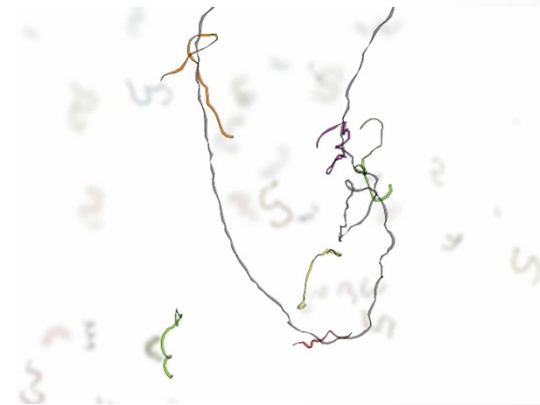
# DNA Origami

- *Folding* long (7000bp) naturally occurring (viral) ssDNA
- By lots of short 'staple' strands that constrain it

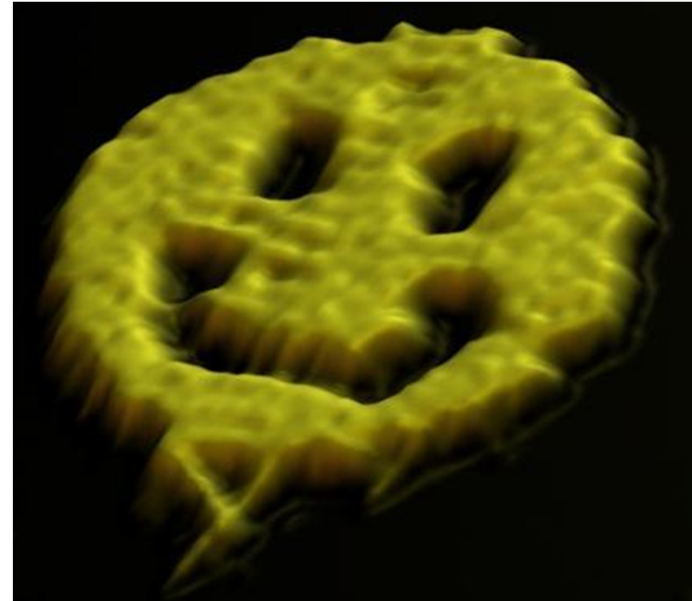
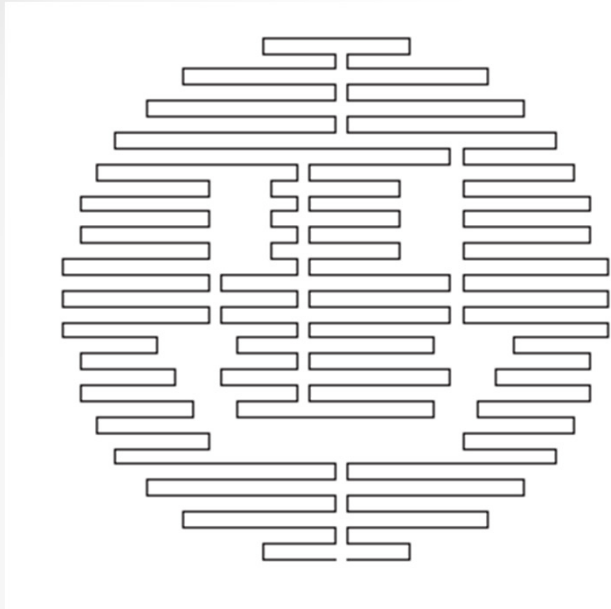


PWK Rothemund, *Nature* 440, 297 (2006)

Black: long viral strand  
Color: short staple strands



# DNA Origami



Paul Rothemund's "Disc with three holes" (2006)

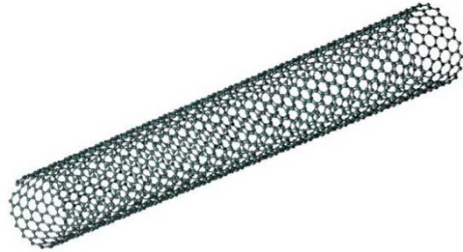


Paul W K Rothemund  
California Institute of Technology

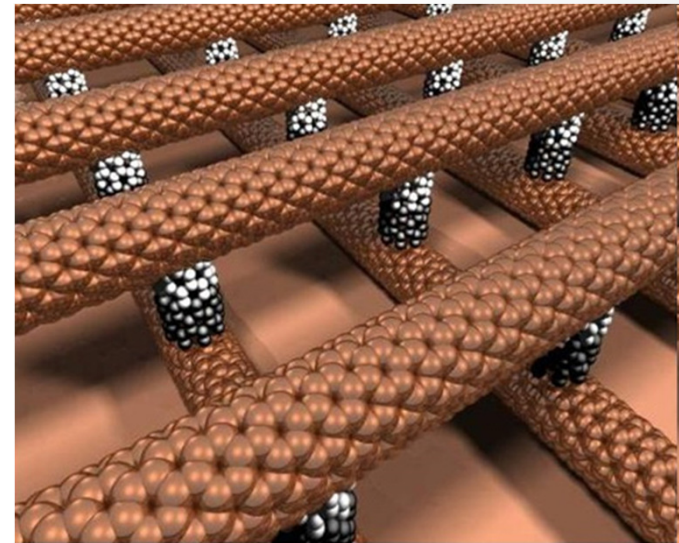
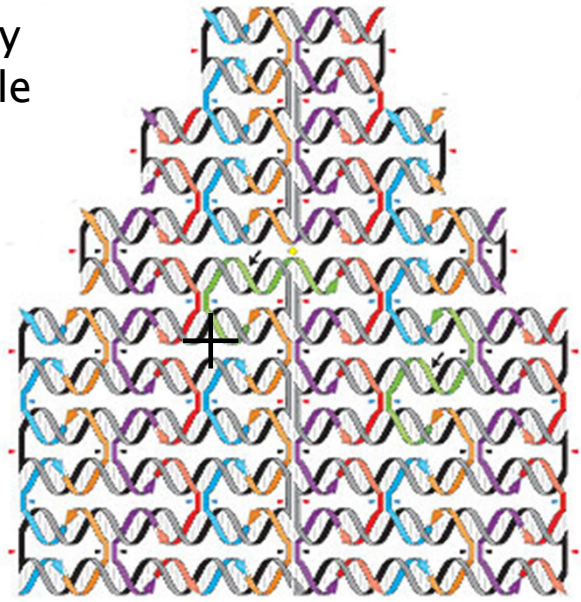
This means we can already self-assemble meso-scale structures.

# DNA Circuit Boards

DNA-wrapped  
nanotubes



6 nm grid of  
individually  
addressable  
pixels

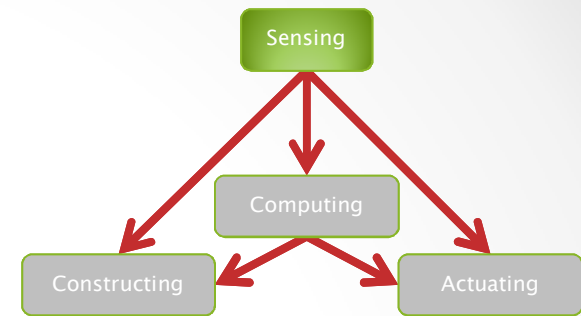


European Nanoelectronics Initiative Advisory Council

"What we are really making  
are tiny DNA circuit boards  
that will be used to  
assemble other  
components."  
*Greg Wallraff, IBM*

PWK Rothemund, *Nature* 440, 297 (2006)



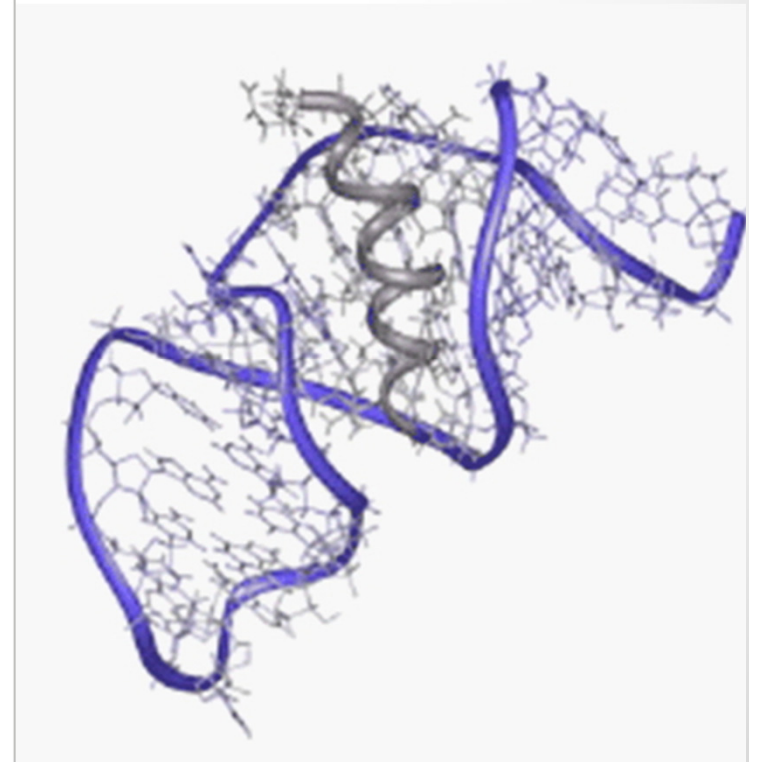
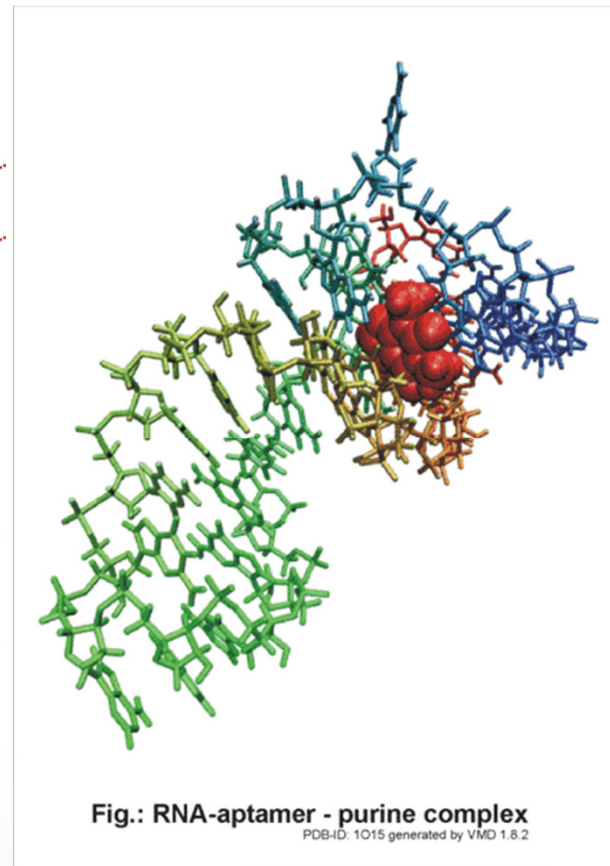
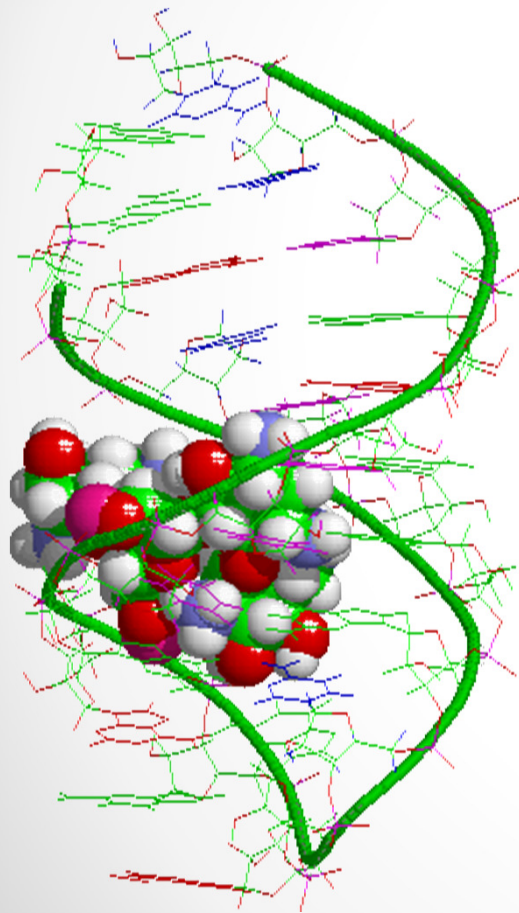


# Sensing

...

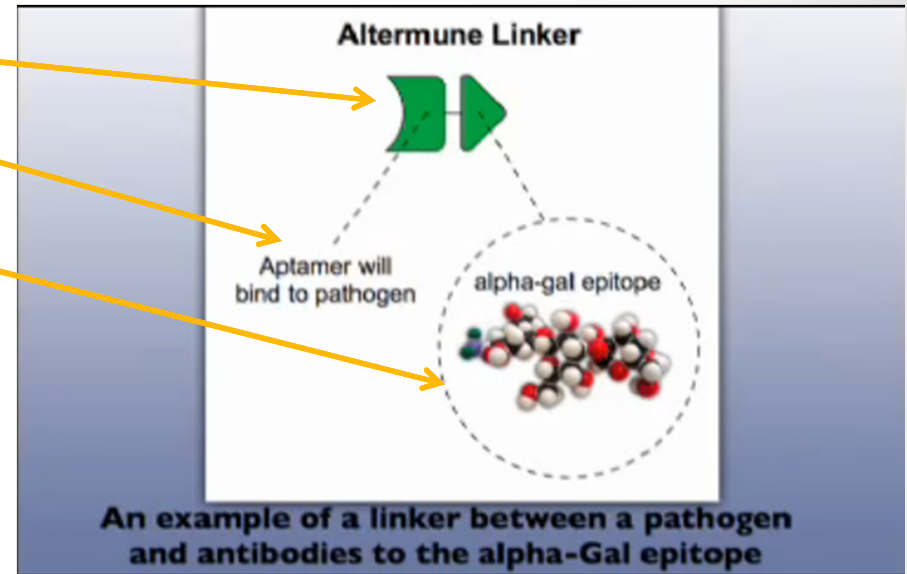
# Aptamers

- Artificially evolved DNA molecules that stick to anything you like (highly selectively).



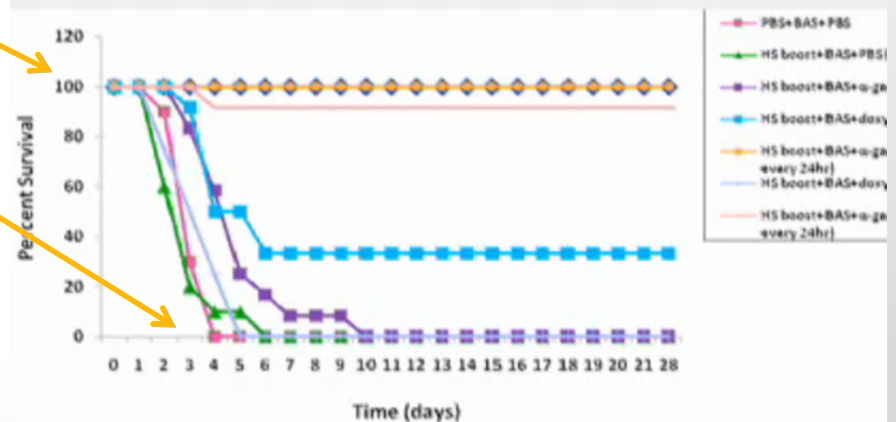
# Pathogen Spotlights

- DNA aptamer binds to:
  - A) a pathogen
  - B) a molecule our immune system already hates and immediately removes (eats) along with anything attached to it



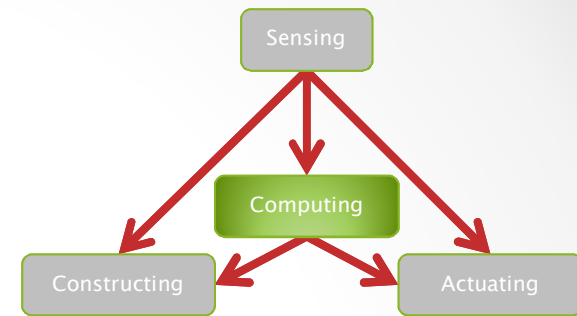
- Result: instant immunity
  - Mice poisoned with Anthrax plus aptamer (100% survival)
  - Mice poisoned with Anthrax (not so good)

*Survival Curve of A/J Mice Immunized with Human Serum, Challenged with BAS and Treated with  $\alpha$ -gal PAA-12 Aptamer and Doxycycline*



Kary Mullis (incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)





# Computing

...

## Basic Steps

# Compositionality

- Sensors and Actuators at the 'edge' of the system
  - They can use disparate kinds of inputs (sensors) and outputs (actuators)
- The 'kernel' of the system computes
  - Must use uniform inputs and outputs
- Compositionality in the kernel
  - Supporting 'arbitrary' computing complexity
  - The **output** of each computing components must be the **same kind of 'signal'** as the **input**
- sdf
  - If the inputs are voltages, the outputs must be voltages
  - If the inputs are DNA, the outputs must be DNA
- Central design question
  - What should our **signals** (not components!) be?
  - Then design components that manipulate those signals.

# Rules of the Game

- Short complementary segments hybridize reversibly



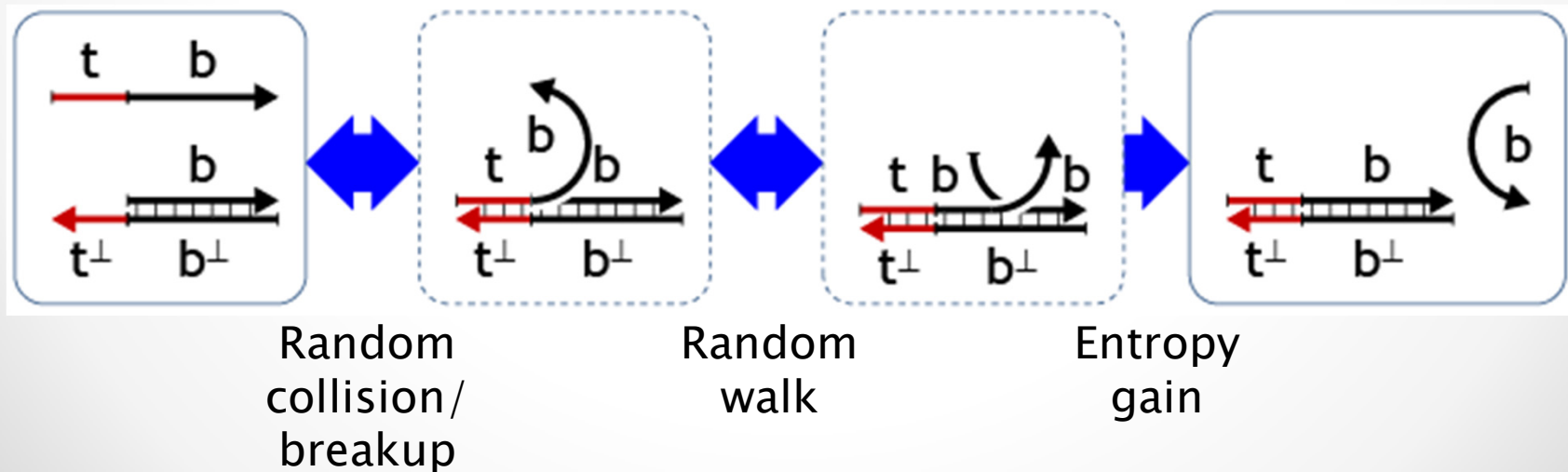
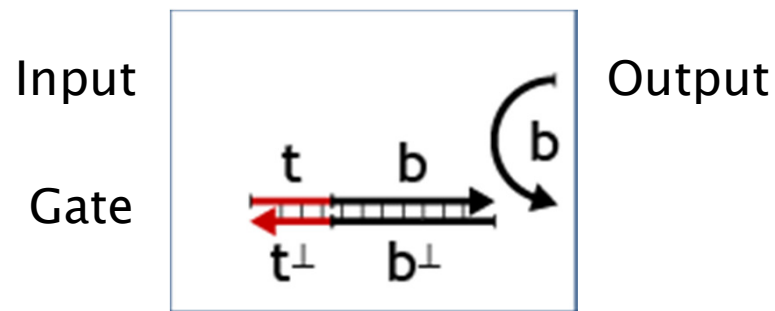
- Long complementary segments hybridize irreversibly





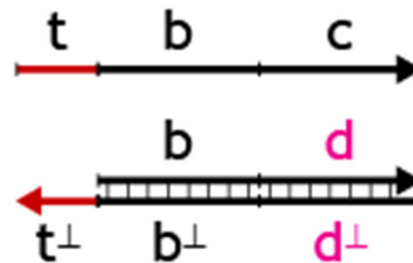
# DNA Strand Displacement

- Short strand (toehold): reversible binding
- Long strand (body): irreversible binding

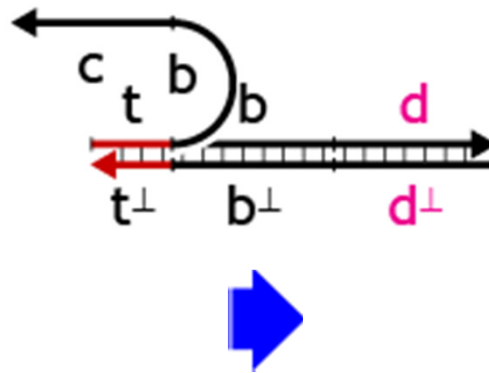


# Failed Strand Displacement

- What if the input does not match the gate?

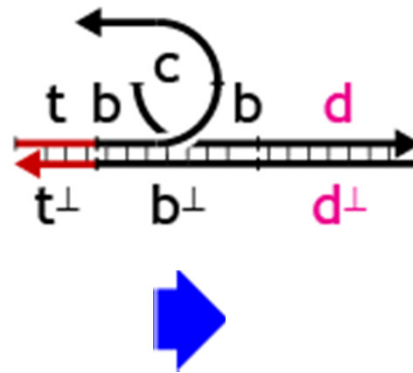


# Failed Strand Displacement

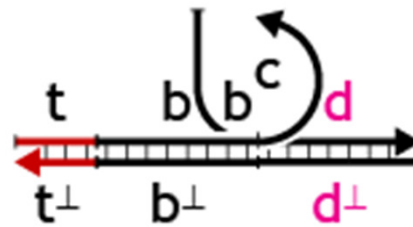




# Failed Strand Displacement

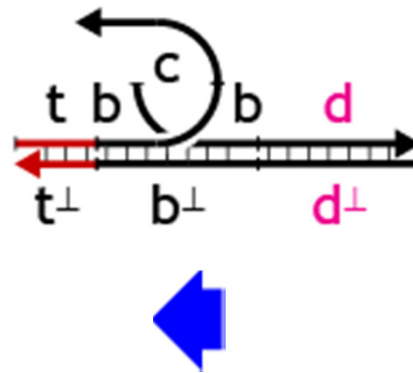


# Failed Strand Displacement

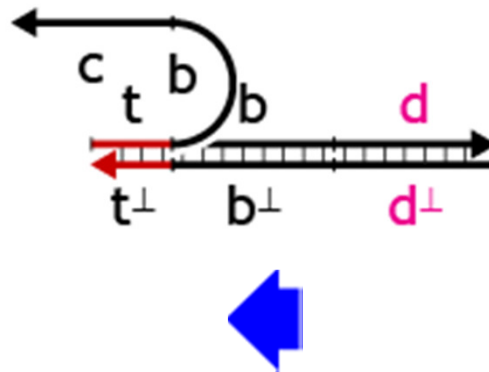


?

# Failed Strand Displacement



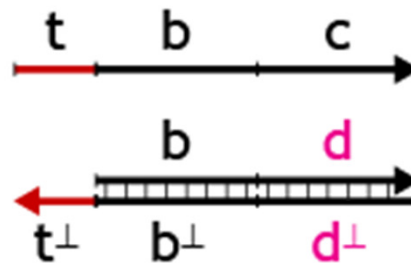
# Failed Strand Displacement



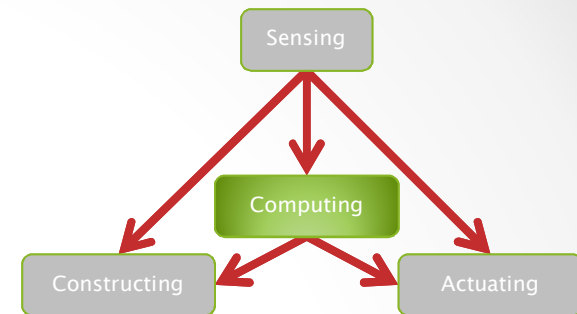


# Failed Strand Displacement

- Hence an incorrect binding will undo
  - That's why toeholds must bind reversibly



- Matching depends on the long segment only
  - Strand displacement succeeds iff the whole long segment matches
  - The address space is determined by the size of the long segment, which is unbounded (not by the size of the toehold)
  - The toehold is just a 'cache' of the address



# Computing

...

Implementing “Arbitrary”  
Computing Functions

# What does DNA Compute?

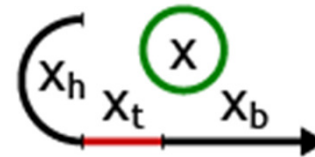
- Electronics has *electrons*
  - All electrons are the same: you can only count them
  - *Few* electrons = **False**; *lots* of electrons = **True**
  - But **Boolean Logic** is only a necessary evil to build symbolic computation
- DNA computing has *symbols* (DNA words)
  - DNA words are not all the same
  - **Symbolic computation on abstract signals** can be done *directly*
  - Signals are presented **concurrently** (in a soup)
  - No requirement to do Boolean Logic
- Then, what are our ‘gates’ (if not Boolean?)
  - Theory of Concurrency
  - Process Algebra as the “Boolean Algebra” of DNA Computing

# Signals

- A signal is the representation of an abstract event
  - E.g. generated by a sensor
  - E.g. accepted by an effector
  - We are not limited to true/false

- 3-domain signals

- $x_h$ : hystory (ignore)
- $x_t$ : toehold (binding)
- $x_b$ : body (recognition)

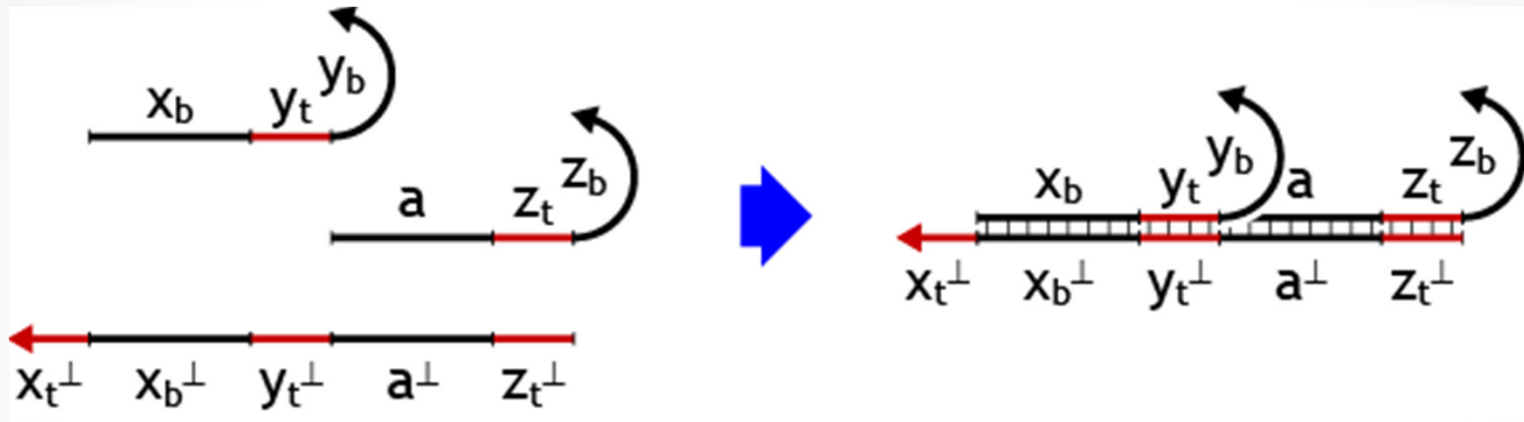


- Signals (single stranded DNA) are prepared by (artificial) **DNA synthesis**



# Gates

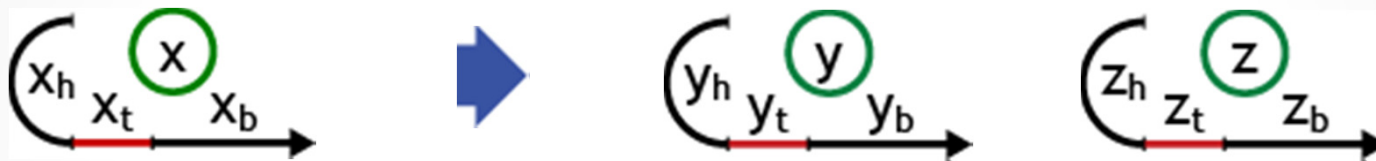
- Double-stranded structures with free toeholds



- Gates are prepared by **self-assembly** from single-stranded DNA that is synthesized

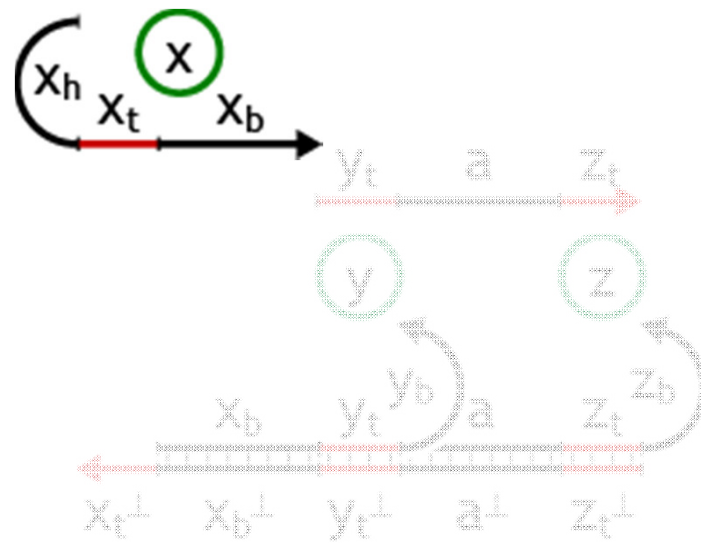
# Fork Gate

- $x \rightarrow y + z$



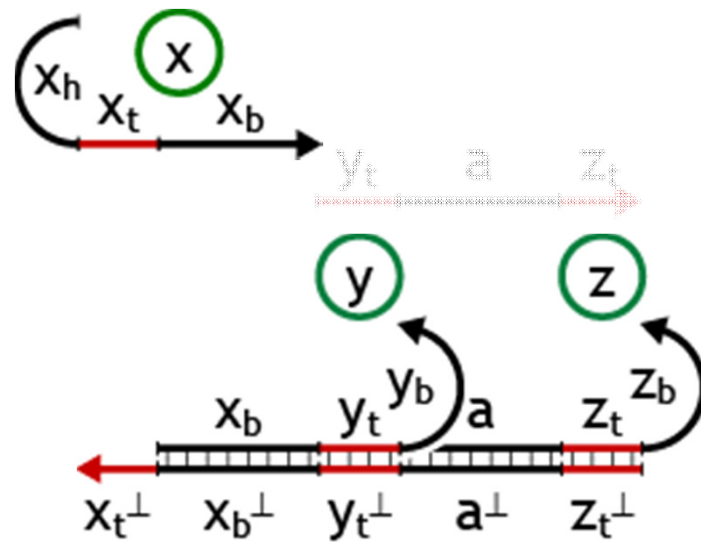
- $x \rightarrow y + 0$  transform  $x$  to  $y$  (transducer)
- $x \rightarrow x + y$  linear production of  $y$  (catalyst)
- $x \rightarrow x + x$  exponential production of  $x$  (amplifier)

# Fork Gate



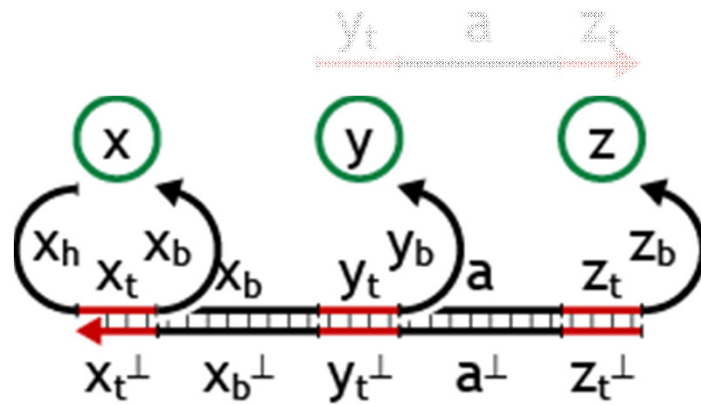
This is the  
Fork Gate  
structure

# Fork Gate

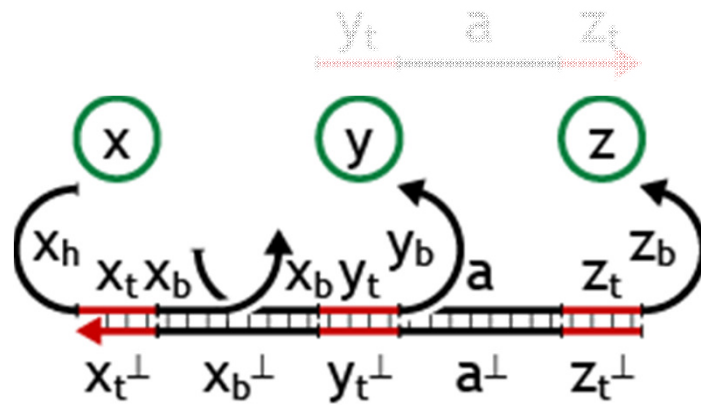




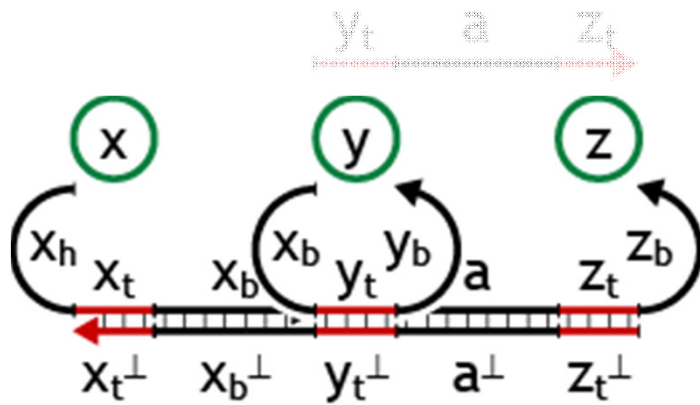
# Fork Gate



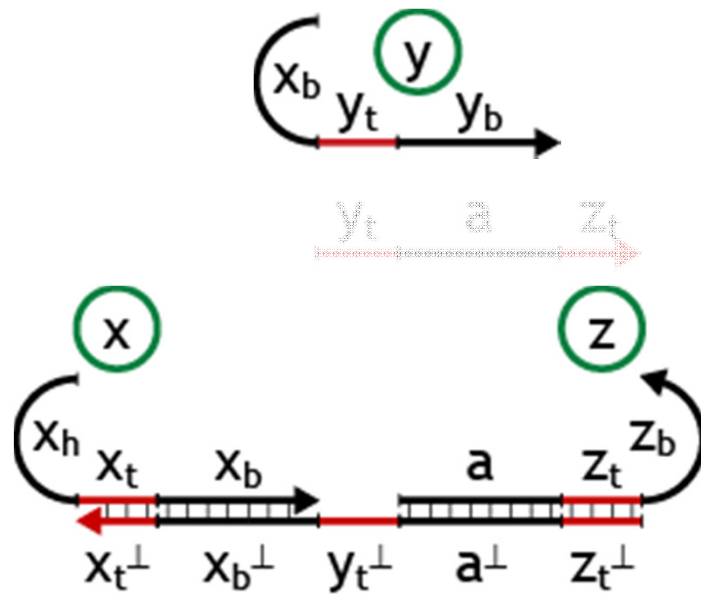
# Fork Gate



# Fork Gate

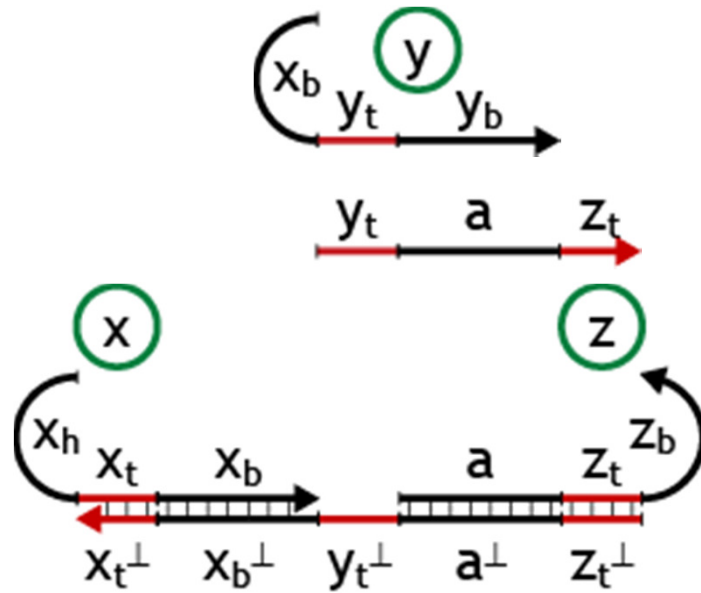


# Fork Gate

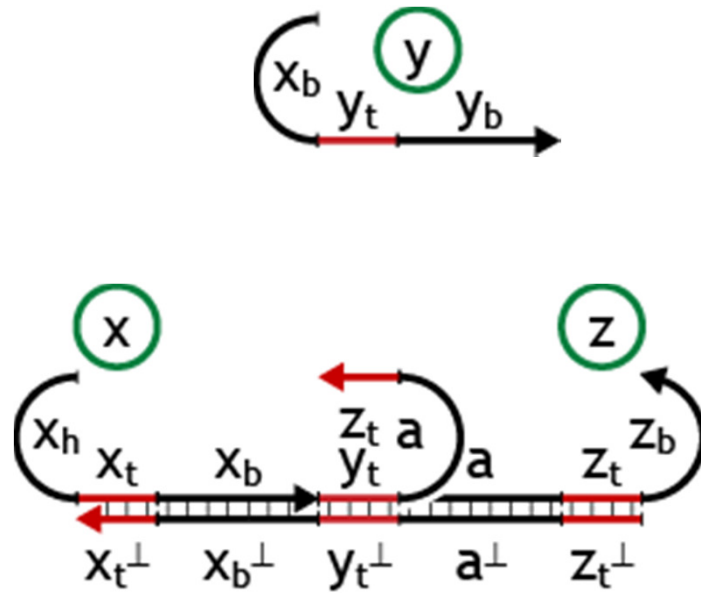




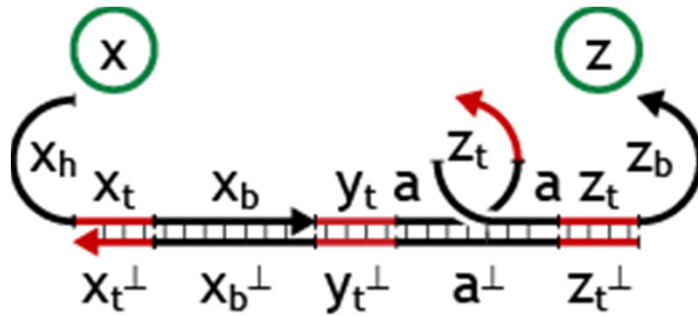
# Fork Gate



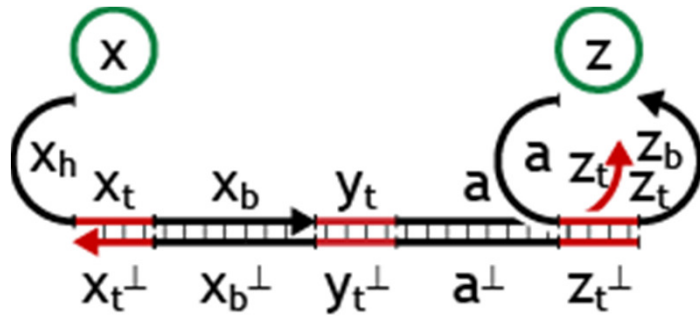
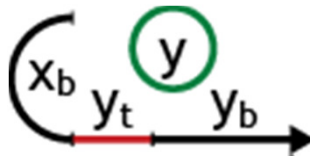
# Fork Gate



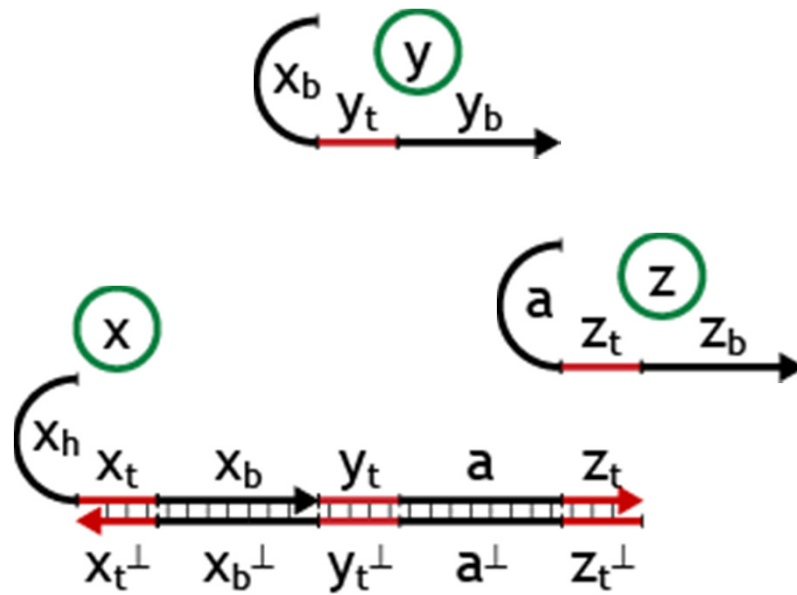
# Fork Gate



# Fork Gate

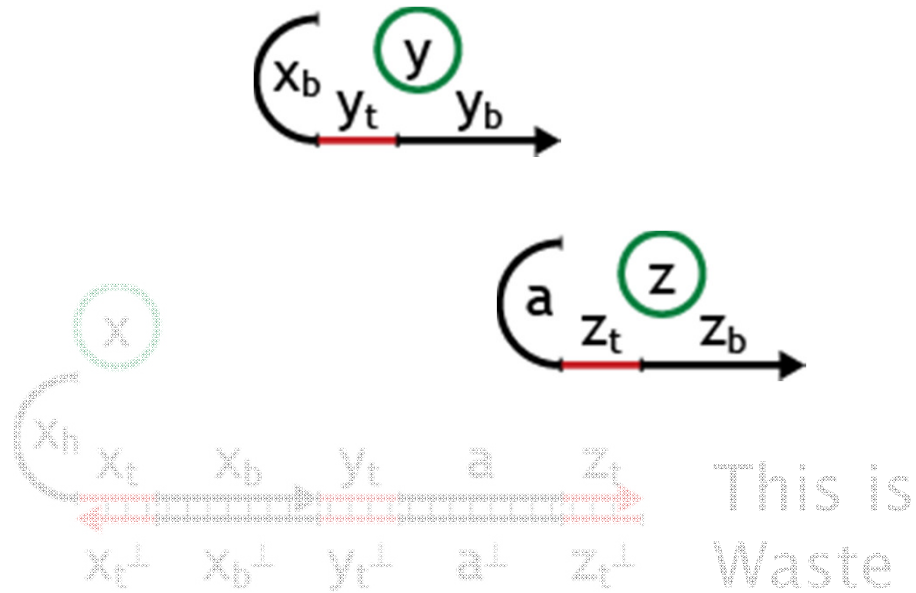


# Fork Gate





# Fork Gate

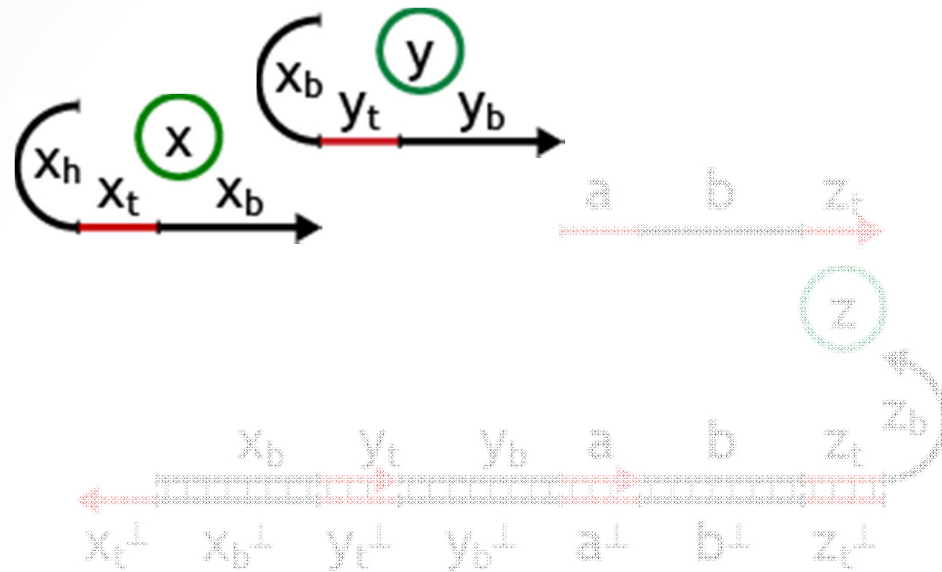


# Join Gate

- $x + y \rightarrow z$

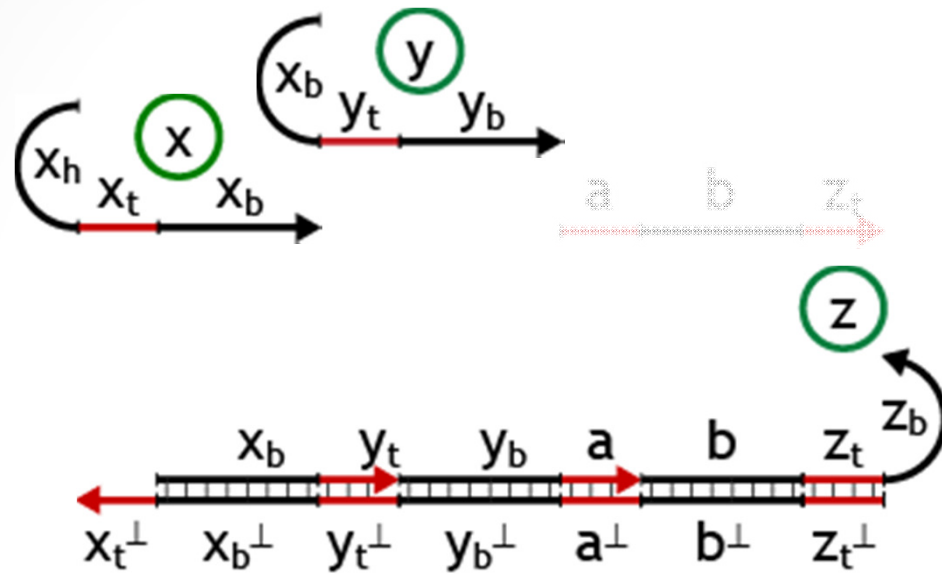


# Join Gate

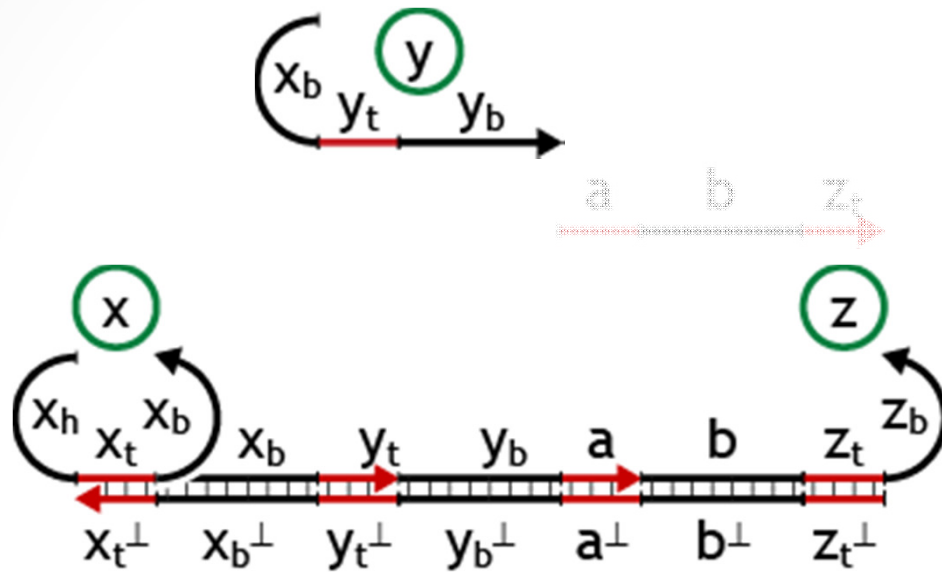


This is the  
Join Gate  
structure

# Join Gate

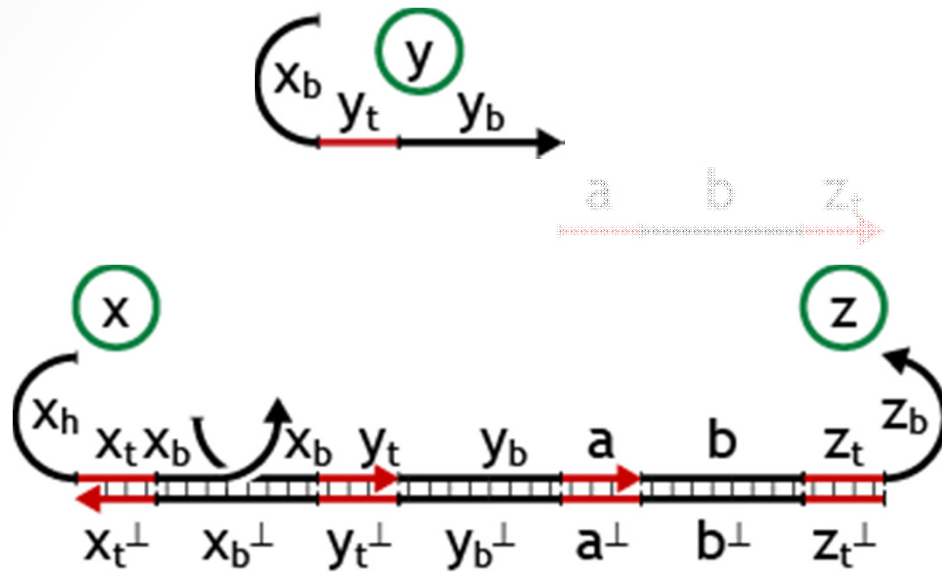


# Join Gate

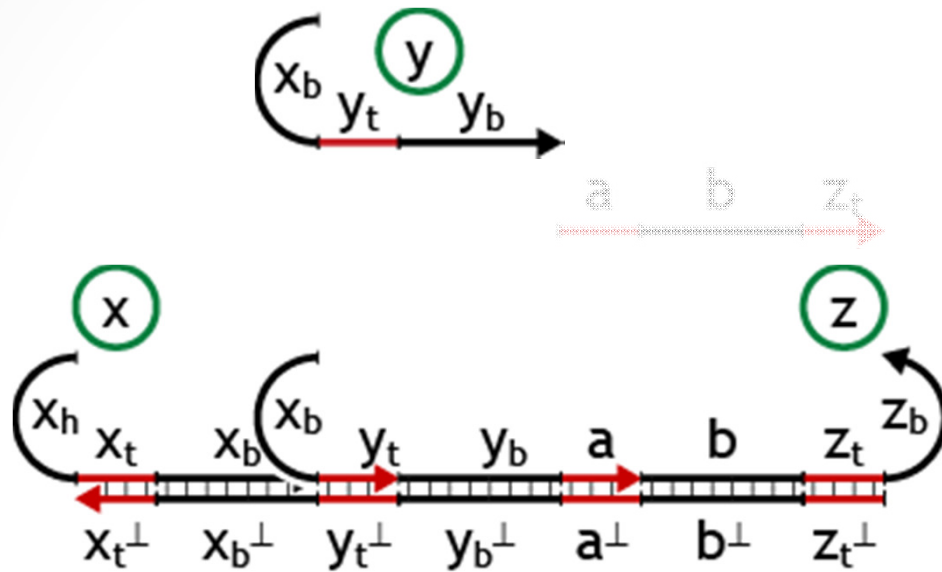




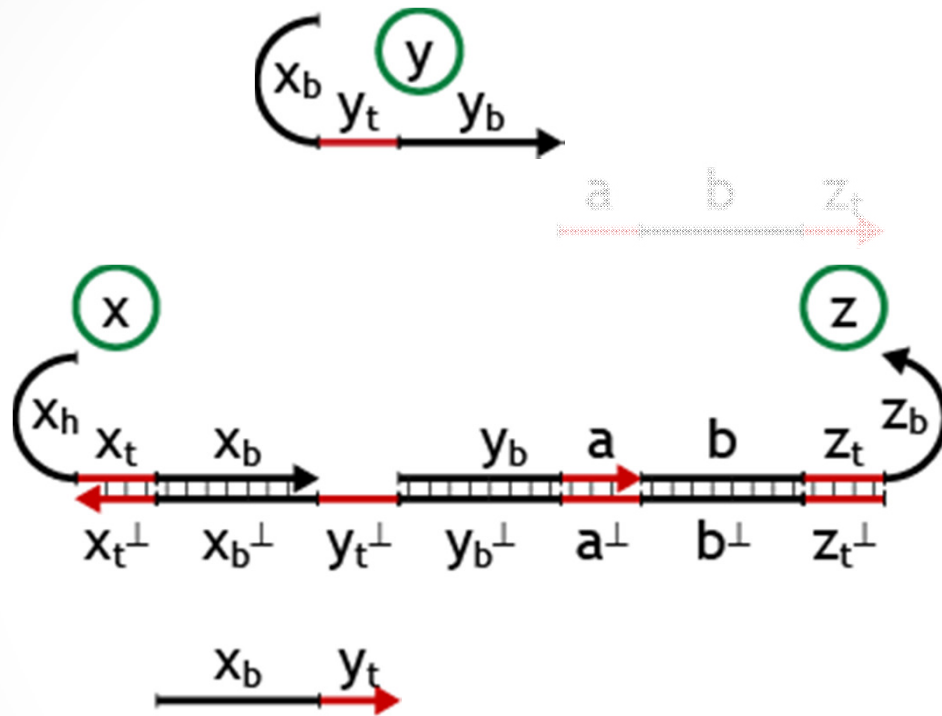
# Join Gate



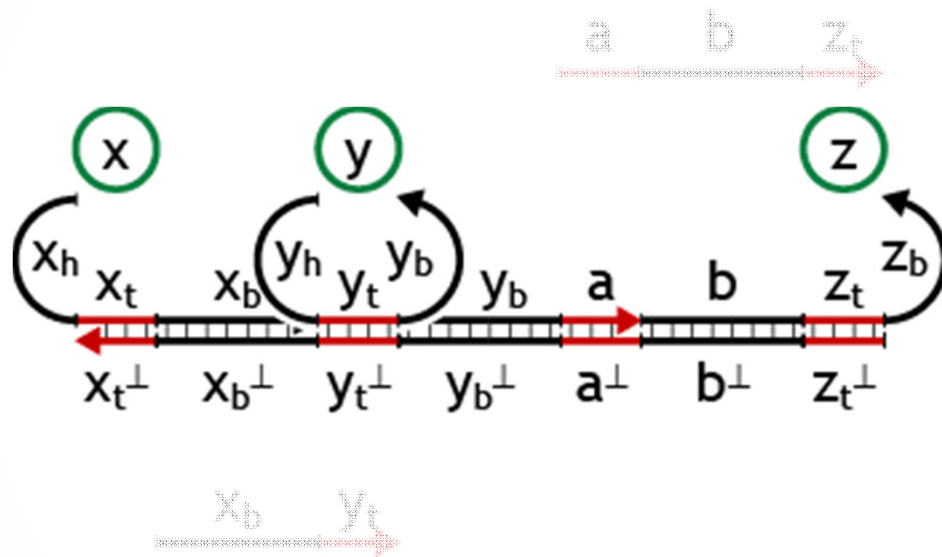
# Join Gate



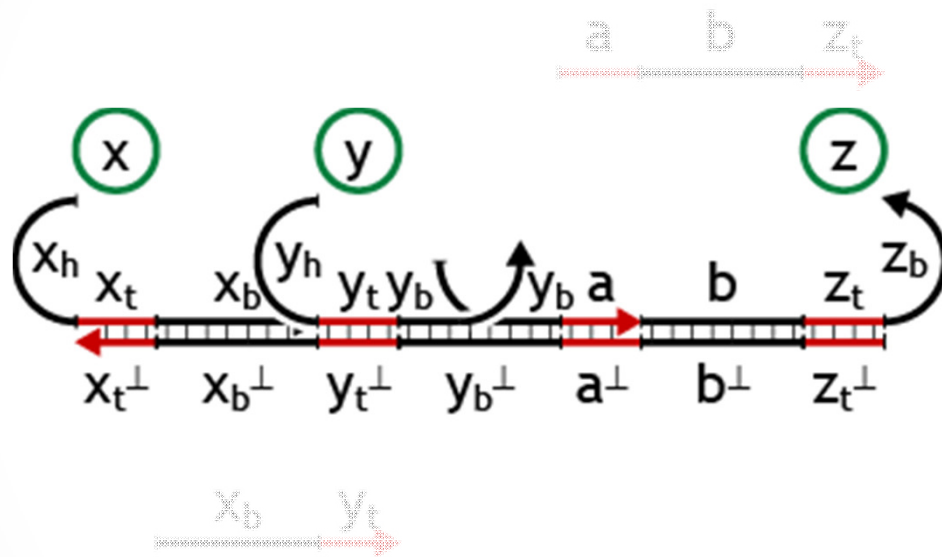
# Join Gate



# Join Gate

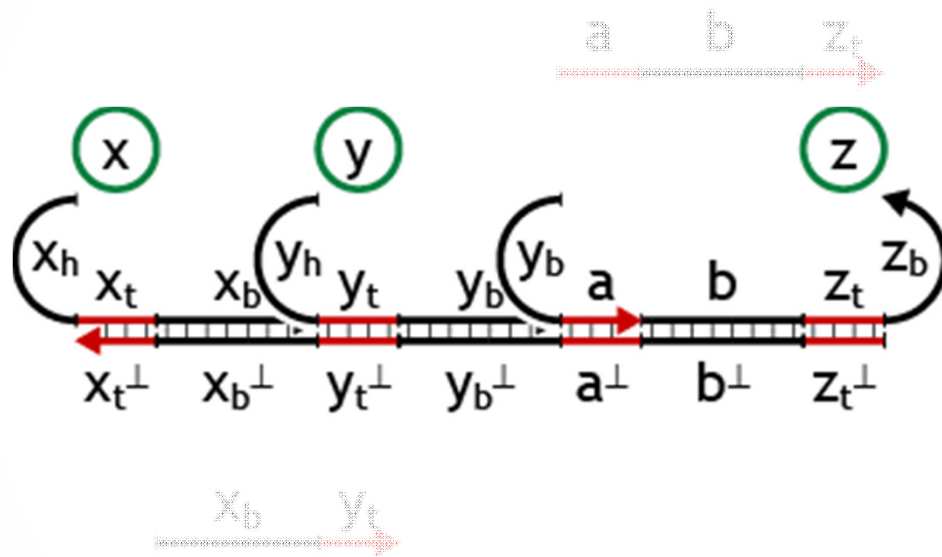


# Join Gate

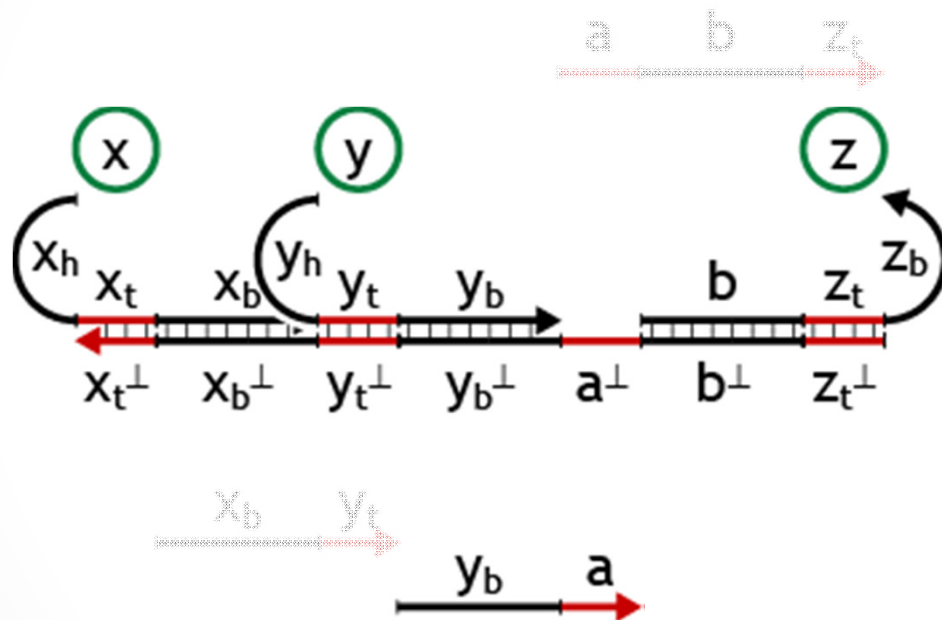




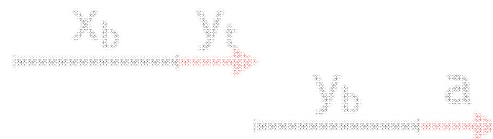
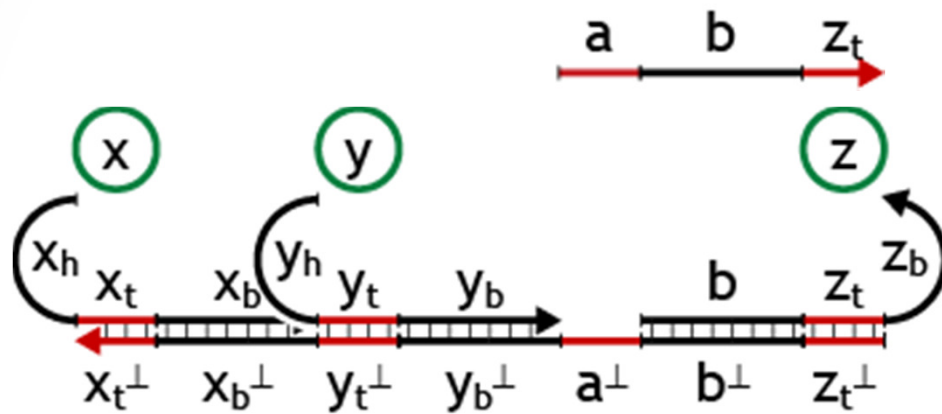
# Join Gate



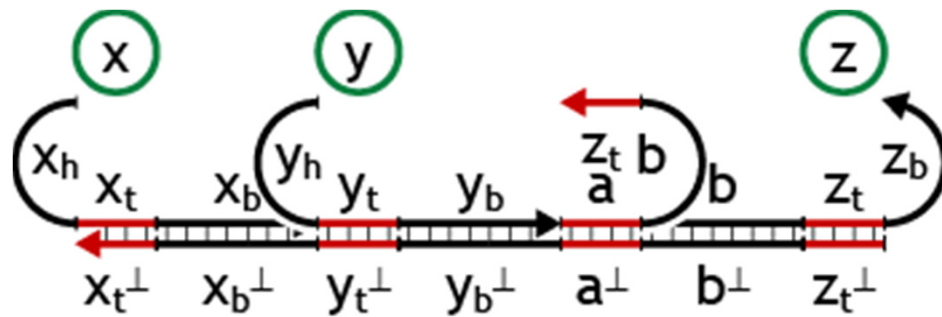
# Join Gate



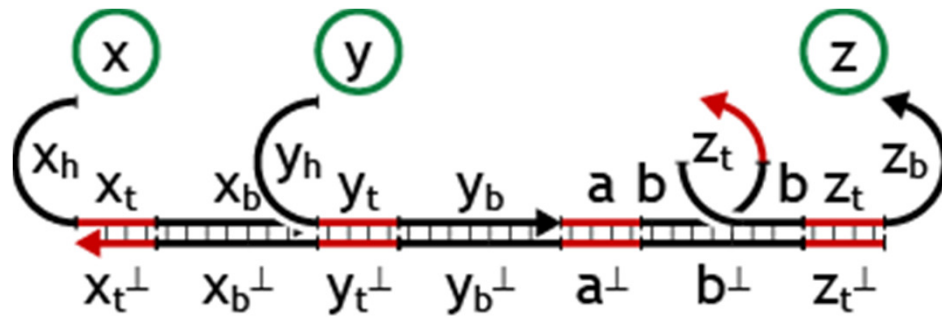
# Join Gate



# Join Gate

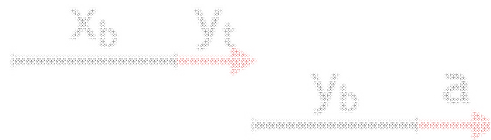
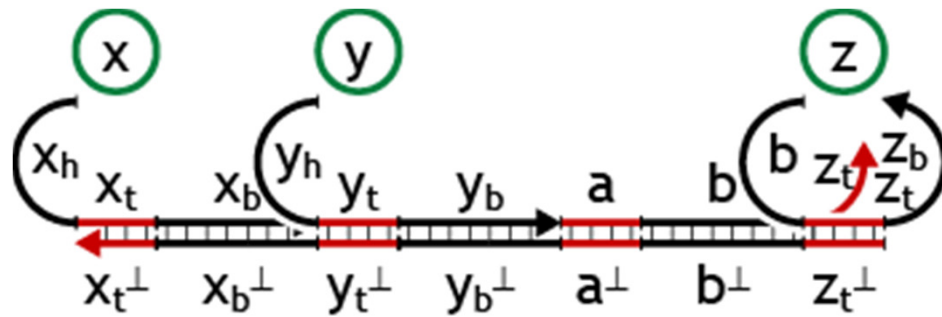


# Join Gate

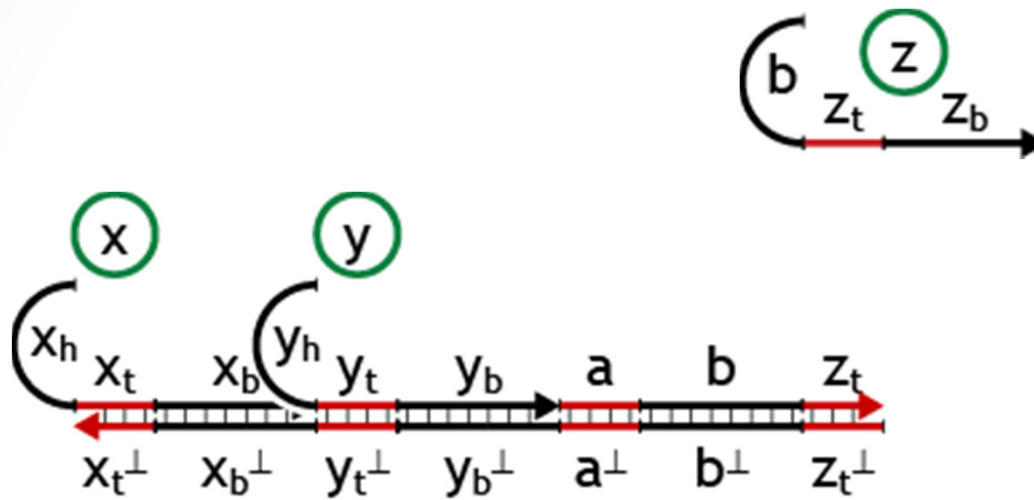




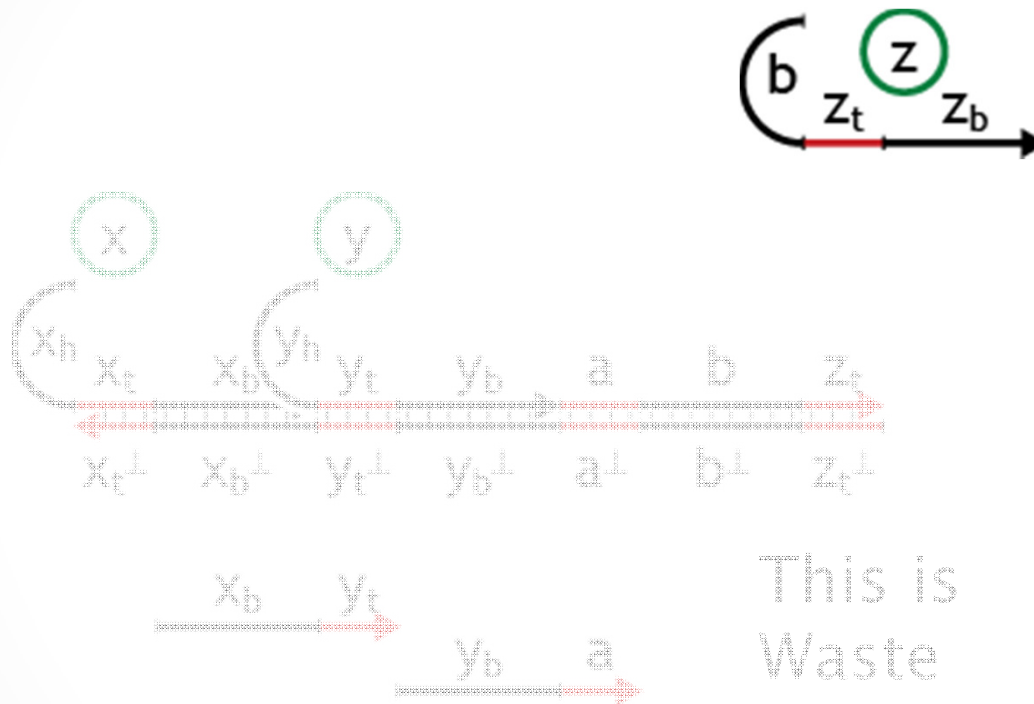
# Join Gate



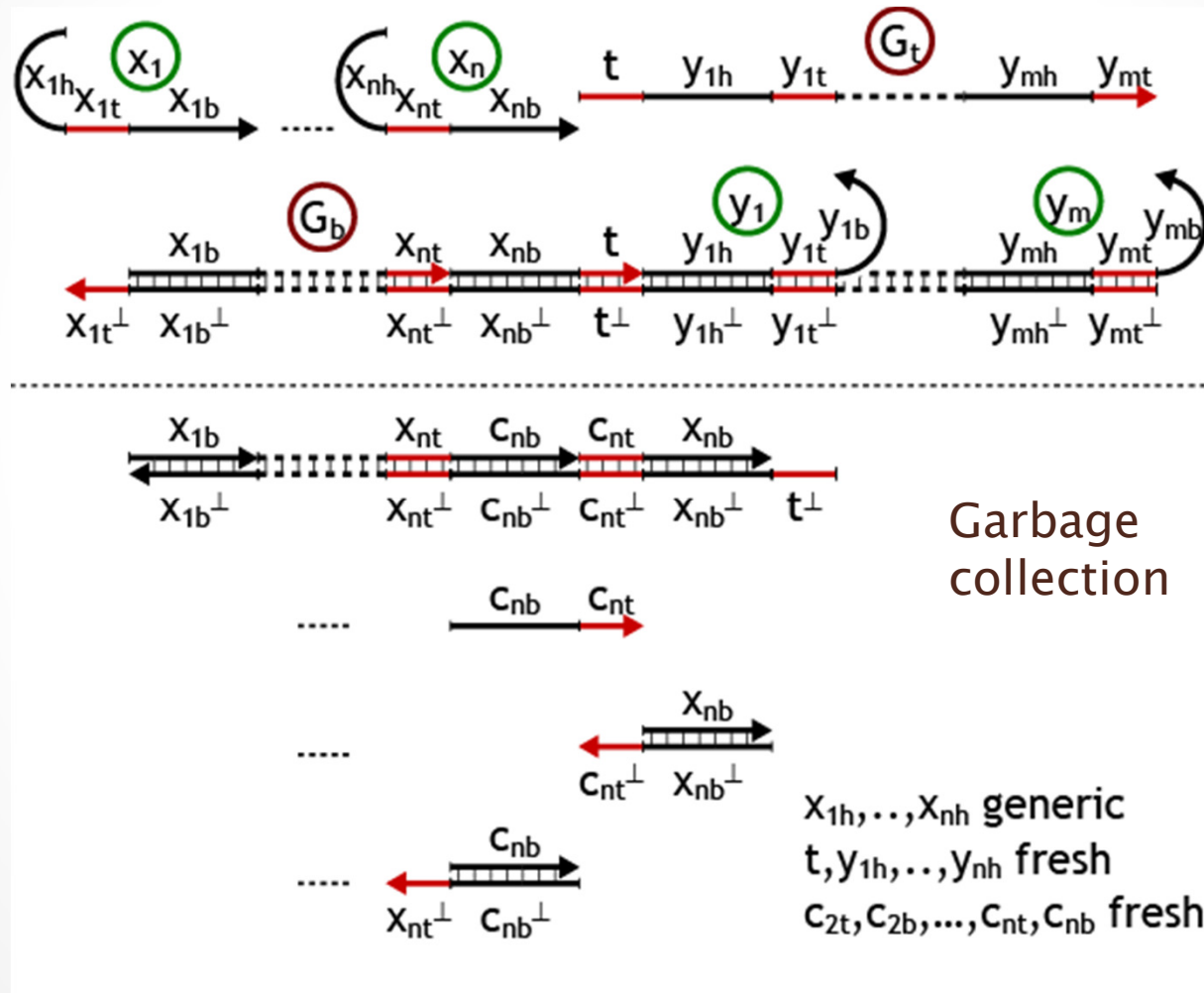
# Join Gate



# Join Gate



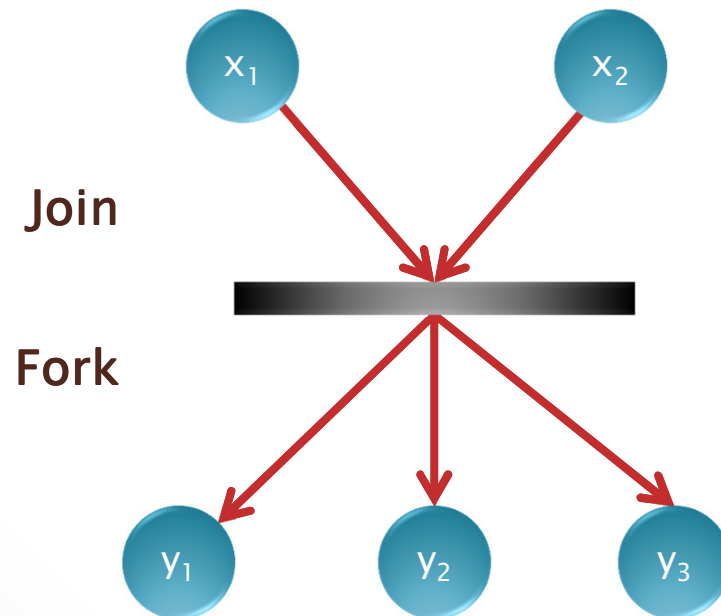
# General n-Join/m-Fork Gate



# Strand Algebra

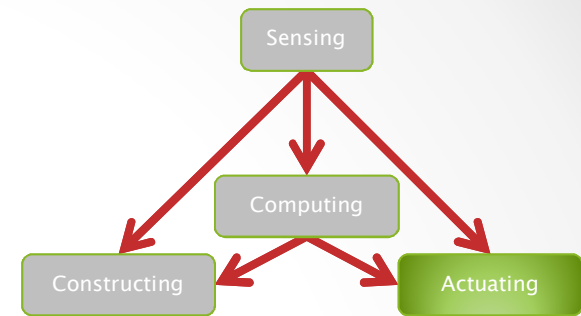
$$x_1 \mid \dots \mid x_n \mid [x_1, \dots, x_n] \cdot [y_1, \dots, y_m] \rightarrow y_1 \mid \dots \mid y_m$$

- Join + Fork + Populations = (Stochastic) Petri Nets



# Gate Design Verification

- Active garbage
  - The active join residuals slow down the performance of following joins.
  - → Add a garbage collector to remove the active residuals.
- Interference between gates
  - The join garbage collector interferes with the fork gate.
  - → Modify the fork gate to remove the interference.
- What else could go wrong?
  - Endless possibilities.
  - → Prove that the fork/join gate structures correctly implement fork/join in all larger circuits.

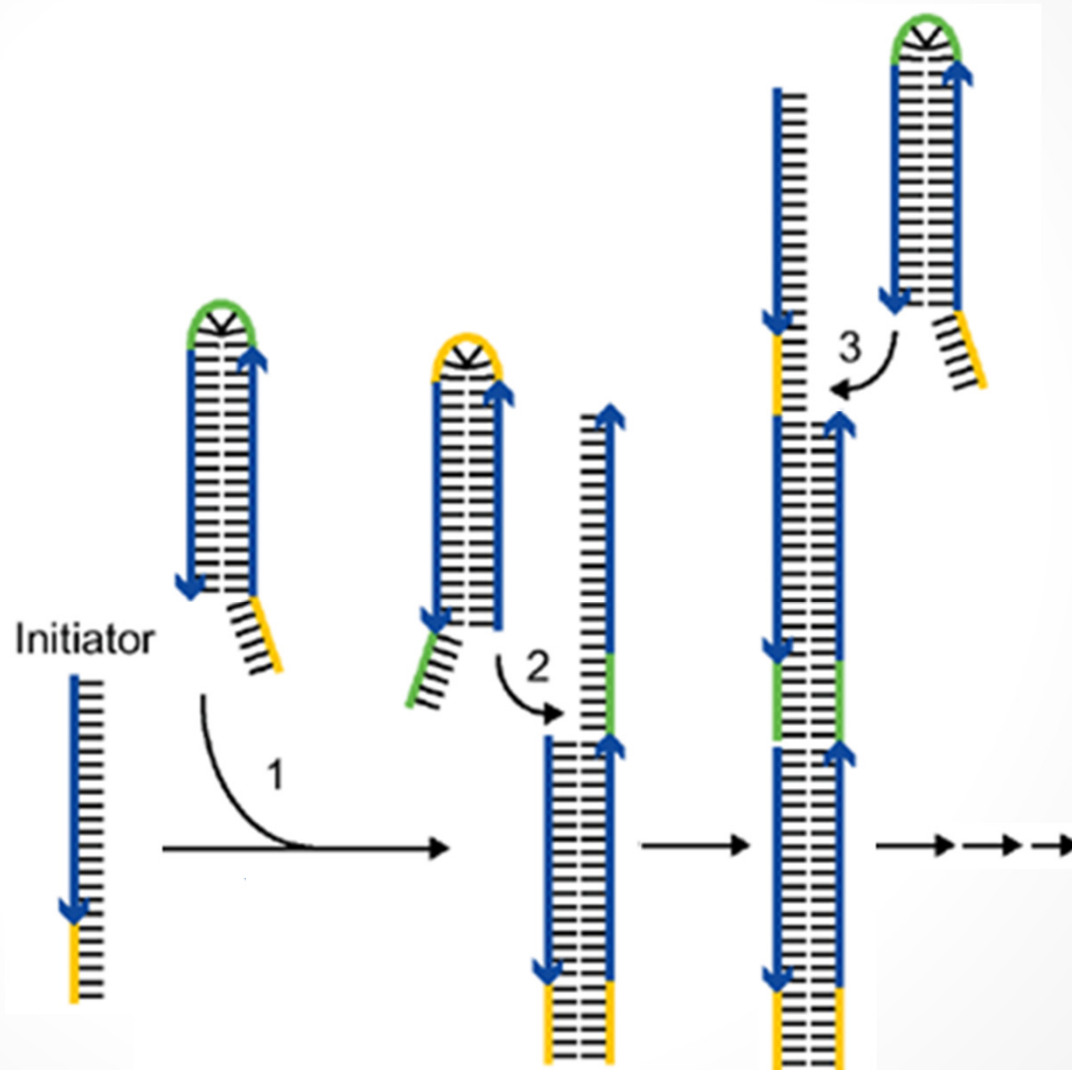
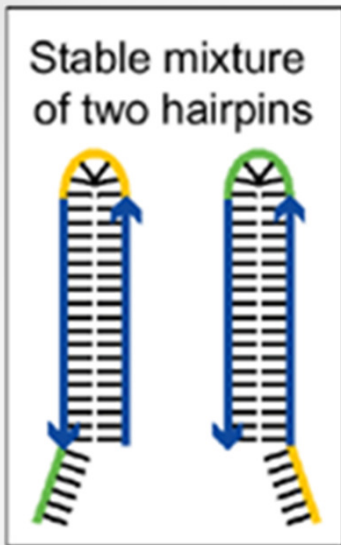


# Actuating

...

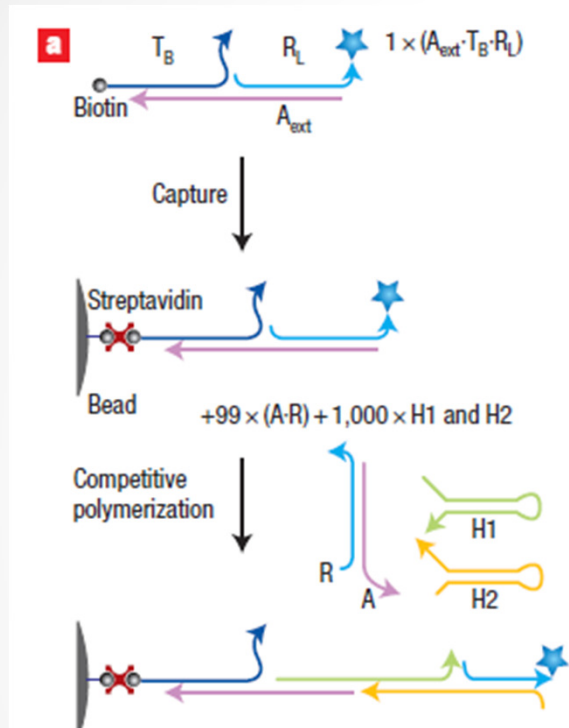


# Hybridization Chain Reaction



chain reaction

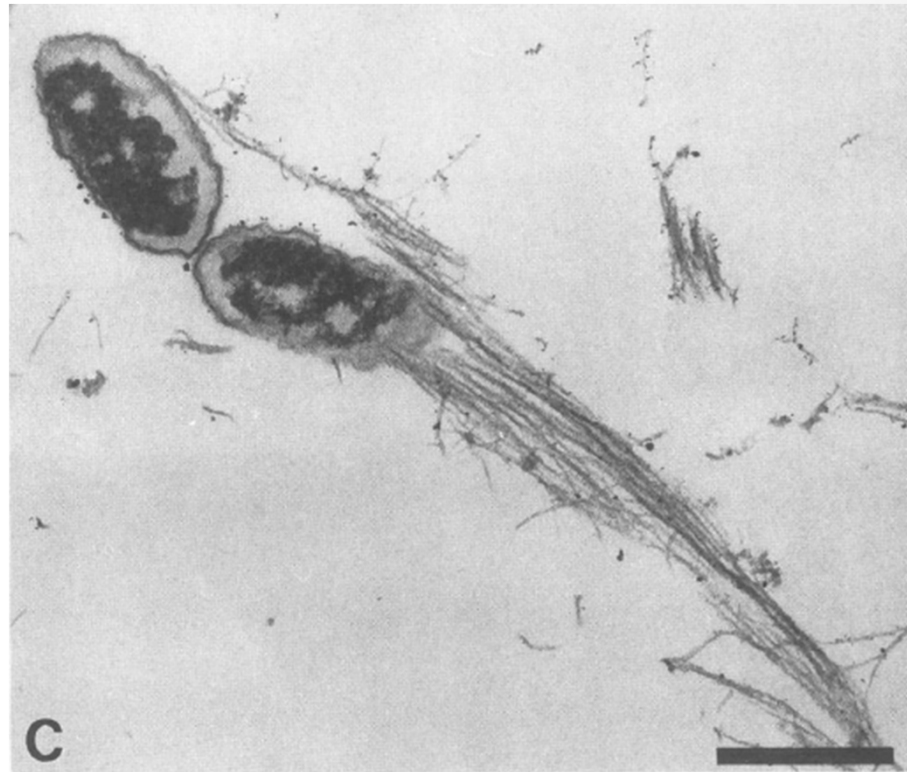
# Polymerization Motor



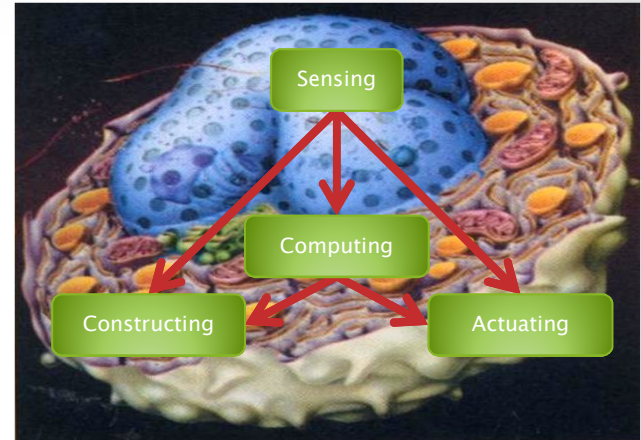
An autonomous polymerization motor powered by DNA hybridization

SUVIR VENKATARAMAN<sup>1</sup>, ROBERT M. DIRKS<sup>1</sup>, PAUL W. K. ROTHMUND<sup>2,3</sup>, ERIK WINFREE<sup>2,3</sup> AND NILES A. PIERCE<sup>1,4\*</sup>

Rickettsia (spotted fever)



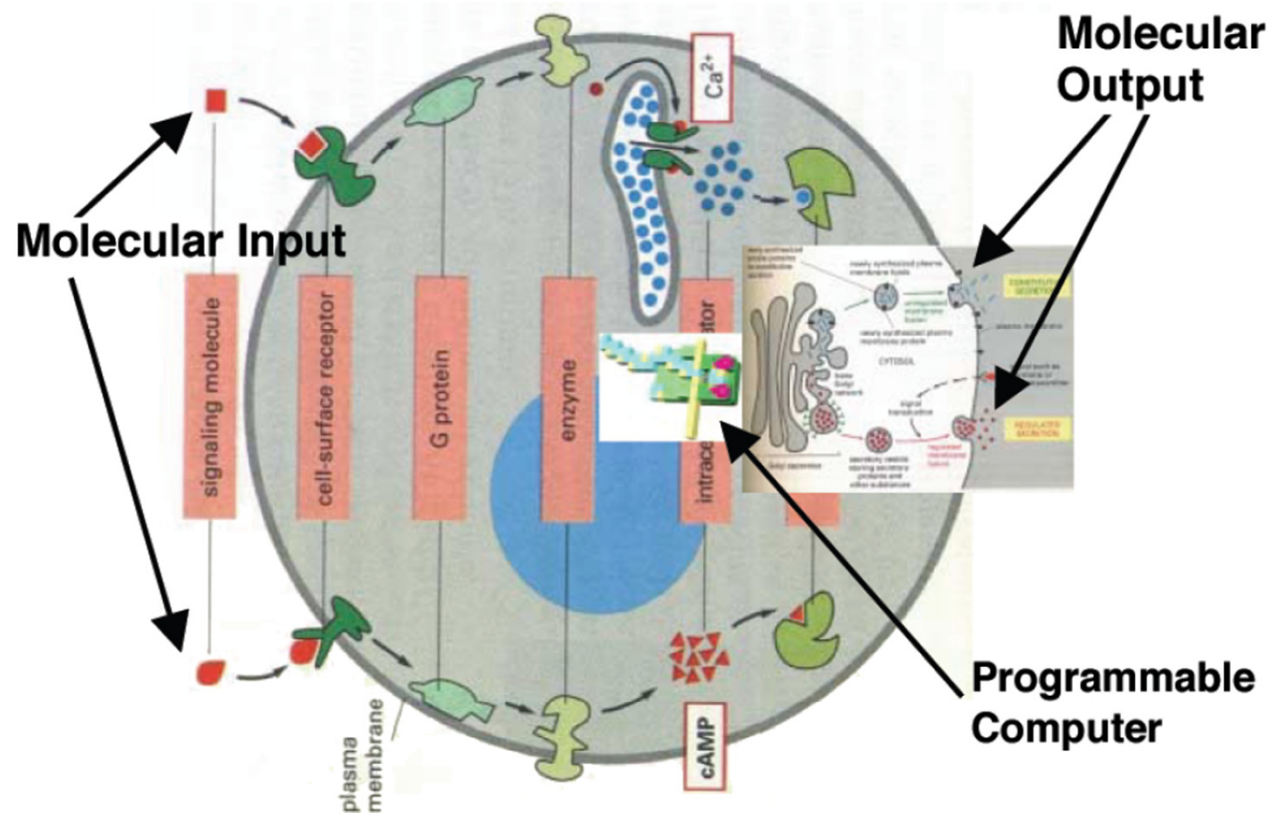
Directional Actin Polymerization Associated with Spotted Fever Group Rickettsia Infection of Vero Cells  
ROBERT A. HEINZEN, STANLEY F. HAYES, MARIUS G. PEACOCK, AND TED HACKSTADT\*



# Curing

...

# A Doctor in Each Cell



*Fig. 1 Medicine in 2050: "Doctor in a Cell"*

Ehud Shapiro

Rivka Adar  
Kobi Benenson  
Gregory Linshitz  
Aviv Regev  
William Silverman

**Molecules and  
computation**

# Tools

...



# Sequence Design

**NUPACK** BETA  
nucleic acid package

Analysis Design Downloads

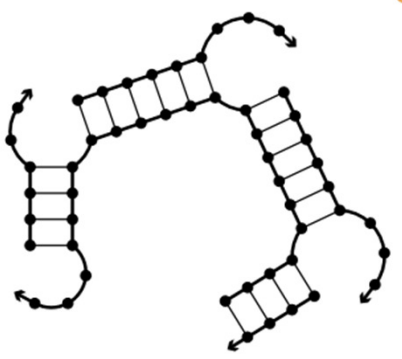
Input References Demos Help

Nucleic acid type:  RNA  DNA

Number of designs: 1

Target structure: (((...+(((...+(((...+(((+)))))))))))))...

Preview:



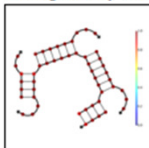
**Input**

**NUPACK** BETA  
nucleic acid package

Analysis Design Downloads

Input Results References Demos Help

Designability summary



Sequence designs

Average percentage of correct nucleotides	Average number of incorrect nucleotides	GC content	Sequence
99.1%	0.475	74.5%	GCCUC+GCAAGCACC+GCC AGCUUG+GCUC+GAGCGCUG GCGCUUGC GCCGUG

Analyze

Copyright © 2007-2009 Caltech. All rights reserved. | [Contact](#) | [Funding](#) | [Terms of use](#)

**Output**

So we can in principle work at this level.

# Visual DSD

## A Strand Displacement Simulator

...

Matthew Lakin, Simon Youssef, Andrew Phillips

<http://lepton.research.microsoft.com/webdna/>



# Syntax

## A programming language for composable DNA circuits

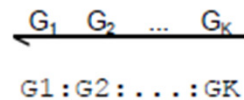
Andrew Phillips\* and Luca Cardelli

### A. Syntax of DNA molecules $D$

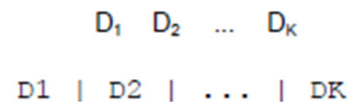
Upper strand with sequence complementary to  $S$



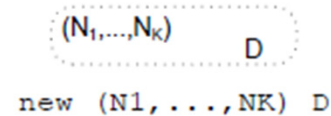
Molecule with segments  $G_1, \dots, G_K$



Parallel molecules  $D_1, \dots, D_K$



Molecules  $D$  with private domains  $N_1, \dots, N_K$

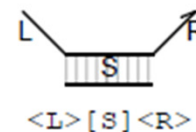


### B. Syntax of DNA segments $G$

Lower strand with toehold  $N^c$

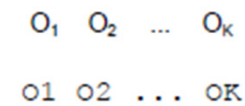


Double strand with sequence  $S$  and overhangs  $L, R$



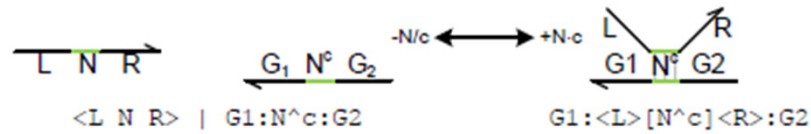
### C. Syntax of DNA sequences $S, L, R$

Sequence of domains  $O_1, \dots, O_K$

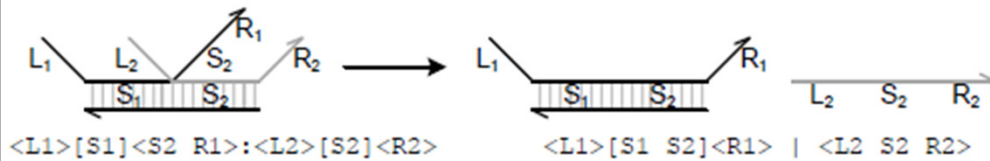


# Dynamics

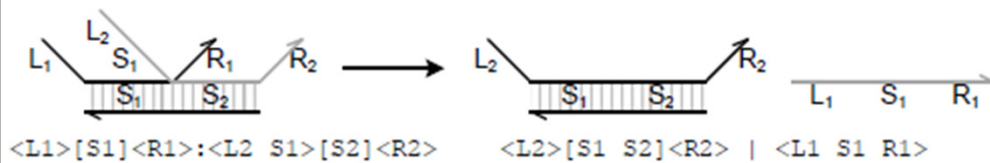
## 1. Toehold binding and unbinding



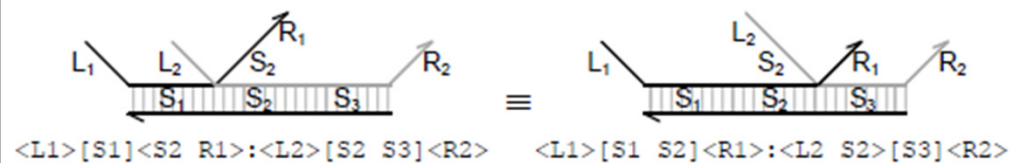
## 2. Strand displacement to the right



## 3. Strand displacement to the left



## 4. Branch migration



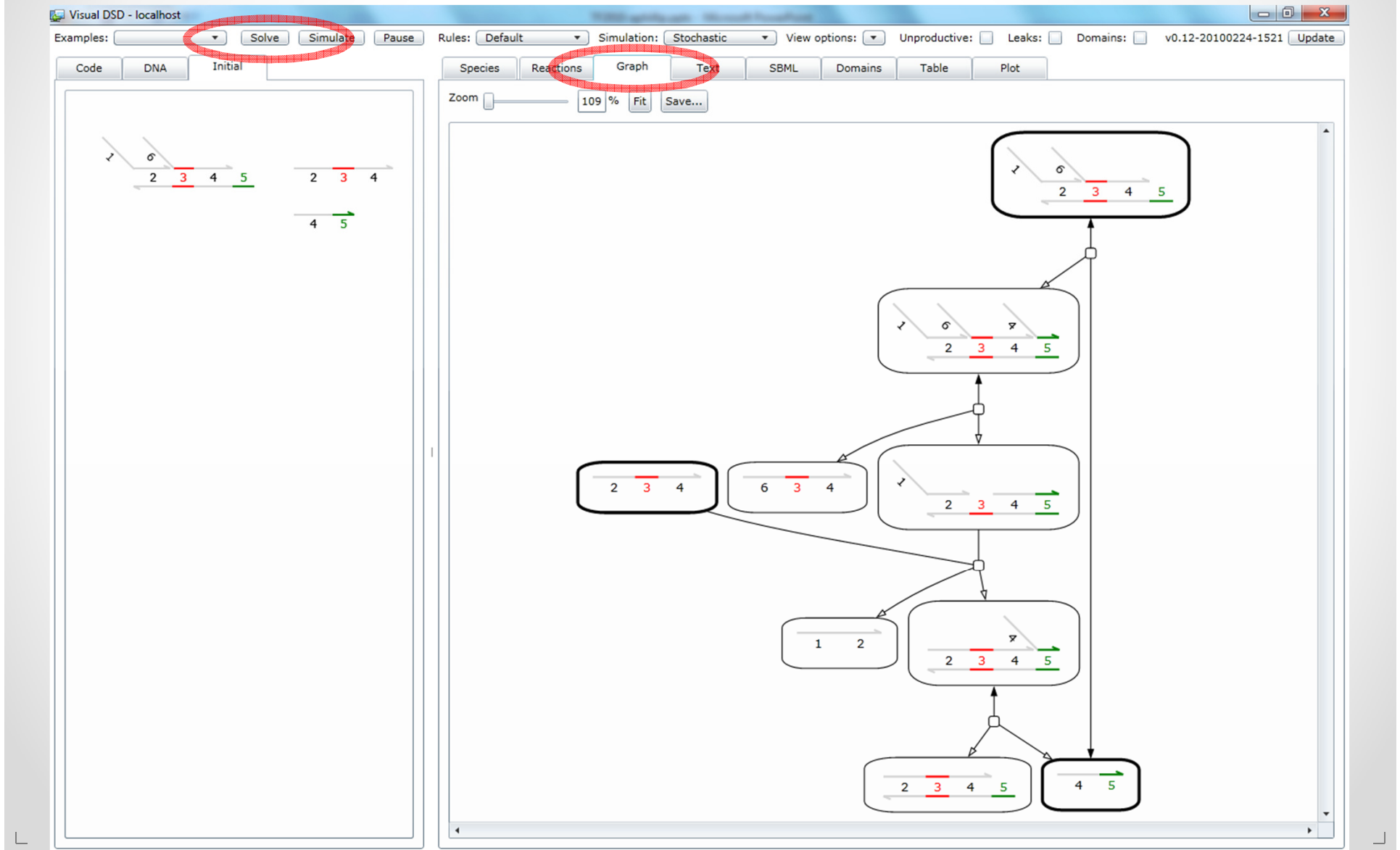
# Initial Species

The screenshot displays the Visual DSD software interface. The window title is "Visual DSD - localhost". The top toolbar includes buttons for "Solve", "Simulate", and "Pause", along with dropdown menus for "Rules" (Default) and "Simulation" (Stochastic). There are also checkboxes for "View options", "Unproductive", "Leaks", and "Domains", and a version number "v0.12-20100224-1521" with an "Update" button.

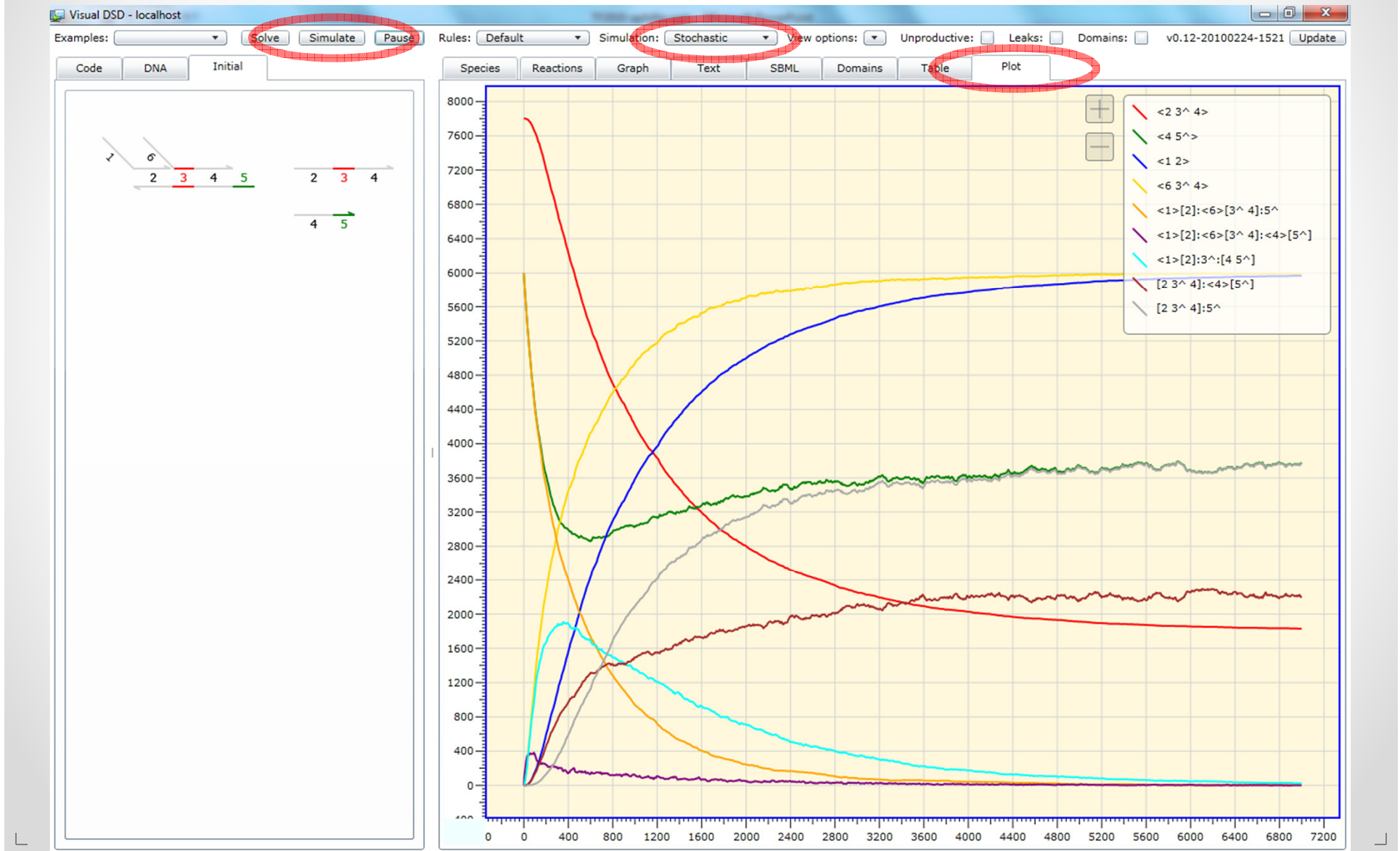
The main interface is divided into two panes. The left pane has tabs for "Code", "DNA", and "Initial". The "Initial" tab is currently selected and highlighted with a red dashed circle. Below the "Initial" tab, a diagram shows a DNA sequence with segments labeled 2, 3, 4, and 5. Segment 3 is red, and segment 5 is green. Arrows indicate the direction of the DNA strands.

The right pane has tabs for "Species", "Reactions", "Graph", "Text", "SBML", "Domains", "Table", and "Plot". The "Graph" tab is selected. Below the tabs, there is a "Zoom" slider set to 109%, and buttons for "Fit" and "Save...". The main area of the right pane is empty.

# Reaction Graph



# Simulation



# Abstract Reactions

Visual DSD - localhost

Examples:  Solve Simulate Pause Rules: **Infinite** Simulation: Stochastic View options: Unproductive:  Leaks:  Domains:  v0.12-20100224-1521 Update

Code DNA Initial

Species Reactions Graph Text SBML Domains Table Plot

Zoom 150% Fit Save...

The diagram illustrates a reaction network for a DNA structure. The nodes represent different states of the DNA, and the arrows represent transitions between these states. The nodes are:

- Top node: DNA structure with segments 2, 3, 4, 5.
- Middle-left node: DNA structure with segments 6, 3, 4.
- Middle-right node: DNA structure with segments 2, 3, 4.
- Bottom-left node: DNA structure with segments 4, 5.
- Bottom-middle node: DNA structure with segments 2, 3, 4, 5.
- Bottom-right node: DNA structure with segments 1, 2.

The transitions are indicated by arrows connecting these nodes, showing the possible reactions between different DNA configurations.

# Detailed Reactions

Visual DSD - localhost

Examples:  Solve Simulate Pause Rules: **Detailed** Simulation: Stochastic View options: Unproductive:  Leaks:  Domains:  v0.12-20100224-1521 Update

Code DNA Initial

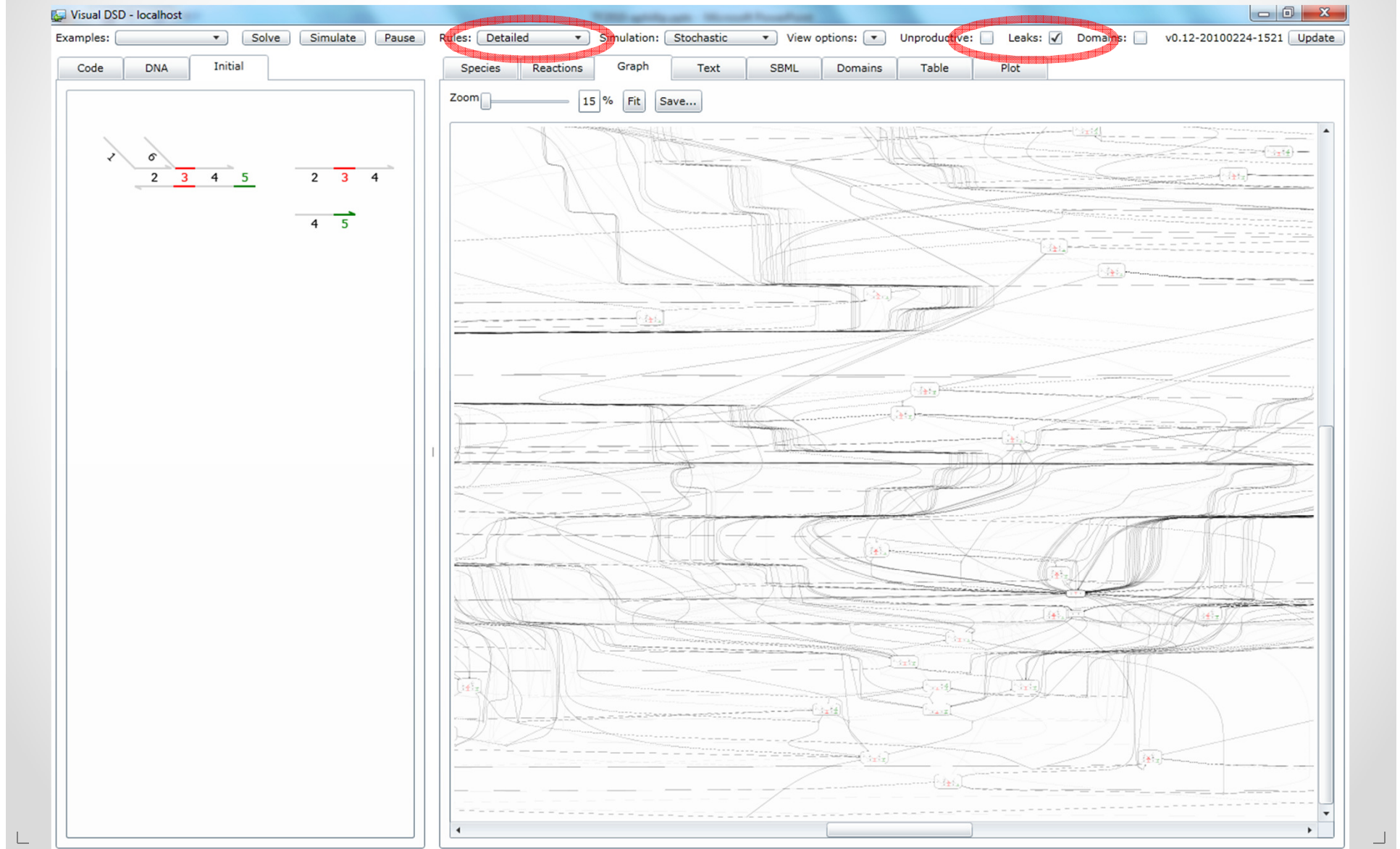
Species Reactions Graph Text SBML Domains Table Plot

Zoom 64% Fit Save...

The screenshot displays the Visual DSD software interface. The title bar reads "Visual DSD - localhost". The top menu bar includes "Examples:", "Solve", "Simulate", "Pause", "Rules: Detailed" (circled in red), "Simulation: Stochastic", "View options:", "Unproductive: ", "Leaks: ", "Domains: ", "v0.12-20100224-1521", and "Update". Below the menu bar is a toolbar with "Code", "DNA", and "Initial" tabs. The main window is divided into two panes. The left pane shows a simplified DNA strand diagram with segments 1-5 and associated reaction arrows. The right pane shows a detailed reaction network diagram with various states of the DNA strand and reaction arrows. The network starts with a single state at the top and branches out into multiple states, eventually leading to a final state at the bottom. The diagram is zoomed in to 64% and includes "Fit" and "Save..." buttons.



# Detailed Leak Reactions!



# Just-in-Time Simulation

Visual DSD - localhost

Examples:  Solve Simulate Pause Rules: **Detailed** Simulation: **JIT** View options:  Unproductive:  Leaks:  Domains:  v0.12-20100224-1521 Update

Code DNA Initial

Species Reactions Graph Text SBML Domains Table Plot

Zoom  38% Fit Save...

The screenshot displays the Visual DSD software interface. At the top, the title bar reads "Visual DSD - localhost". Below it, a control bar contains several buttons: "Examples:", "Solve", "Simulate", "Pause", "Rules: Detailed", "Simulation: JIT", "View options:", "Unproductive: ", "Leaks: ", "Domains: ", "v0.12-20100224-1521", and "Update". The "Rules: Detailed", "Simulation: JIT", and "Leaks: " elements are circled in red. Below the control bar, there are tabs for "Code", "DNA", and "Initial". The "DNA" tab is active, showing a diagram of DNA sequences with colored segments (red, green). The main area of the interface is a large window displaying a complex reaction network graph. The graph consists of numerous nodes and edges, with nodes containing small diagrams of DNA sequences. The graph is zoomed to 38%. The interface also includes a "Zoom" slider, a "Fit" button, and a "Save..." button.

# DNA Sequences

The screenshot displays the Visual DSD software interface. At the top, the title bar reads "Visual DSD - localhost". Below the title bar, there is a menu bar with options: "Solve", "Simulate", "Pause", "Rules: Default", "Simulation: Stochastic", "View options:", "Unproductive:", "Leaks:", "Domains:", "v0.12-20100302-1033", and "Update".

The main interface is divided into several sections:

- Code**: A tabbed interface with "Code", "DNA", and "Initial" tabs. The "Code" tab is active, showing a list of sequences under "TOEHOLD SEQUENCES" and "SPECIFICITY SEQUENCES".
- Check sequences**: A checkbox labeled "Check sequences" is checked, with a "Reset" button next to it.
- TOEHOLD SEQUENCES**: A list of 16 DNA sequences:

```
TATTCC
GCTA
GTCA
TACCAA
CATCG
ACTACAC
CTCAG
CTCAATC
CCTACG
TCTCCA
CCCT
GACA
ACCT
TAGCCA
CACACA
AGAC
```
- SPECIFICITY SEQUENCES**: A scrollable list of 24 DNA sequences:

```
CCCAAAACAAAACAAAACAA
CCCTTTTCTAAACTAAACAA
CCCTTTACATTACATAACAA
CCCTTATCATATCAATACAA
CCCTTAACTTAAACAAATCTA
CCCTATTCAATTCAAATCAA
CCCTATACTATACAATACTA
CCCTAATCTAATCATAACTA
CCCTAAACTTATCTAAACAT
CCCATTTCAAATCAAACCTT
CCCATTTCTAATCAATTCAA
CCCATATCTATACATTACAA
CCCATAACTATTCTAAACTA
CCCAATTCTTAAACATACAA
CCCAATACTATTCTAAACAT
CCCAATCTTAACTATACTA
CCTATACCTTAACTTAAACAA
CCATATCCATAACTTTACAA
CCATAACCTATACTTACAA
CCATTTCCCTTTCTTAACTA
CCATTACCATATCTTATCAT
CCAAAACCATAAACAACCTT
```
- Species**: A tabbed interface with "Species", "Reactions", "Graph", "Text", "SEML", "Domains", "Table", and "Plot" tabs. The "SEML" tab is highlighted with a red circle.
- Zoom**: A slider control for zooming in and out of the main display area.
- Main Display Area**: A large text area showing the following sequences:

```
3^ --> TATTCC
5^ --> GCTA
1 --> CCCTTTACATTACATAACAA
2 --> CCCAAAACAAAACAAAACAA
4 --> CCCTTTTCTAAACTAAACAA
6 --> CCCTTATCATATCAATACAA
```

# Final DNA Circuit

Visual DSD - localhost

Examples:    Rules:  Simulation:  View options:  Unproductive:  Leaks:  Domains:  v0.12-20100302-1520

Code DNA Initial

Species Reactions Graph Text SBML Domains Table Plot

Zoom

The screenshot shows the Visual DSD software interface. The left panel, titled 'Initial', displays the DNA components: a double-stranded DNA molecule with sticky ends (GGGAAATGTAATGATTGTT and GGGAAATAGTATAGTTATGTT) and single-stranded DNA molecules (GGGTTTGGTTTGGTTTGGTT, GGGAAAAGATTGGATTGTT, and GGGAAAAGATTGGATTGTT). The right panel, titled 'Graph', shows a hierarchical assembly graph of the circuit. A red button labeled 'Place Order' is overlaid on the top left of the graph area. The graph shows the assembly of the DNA components into a final structure, with nodes representing different stages of the assembly process.

# Next-Day Oligos!

**XX-IDT**  
INTEGRATED DNA  
TECHNOLOGIES

Chat is now closed.  
Please click to email  
a representative.

[Logout]  
Spain

Item €1,00

Home Products Order Support Services SciTools Search Go

**SameDay® Oligo Service**

**Only € 1,44 EUR / Base!**

The Current Time is 22:40 (GMT)

**Specifications:**

- Order online by 11:00 GMT
- SameDay® priority shipping for delivery by 10:30 GMT on second business day
- 2-OD minimum guarantee (sufficient for > 250 PCR reactions)
- 15-45 bases
- Shipped lyophilized in tubes
- Deprotected & desalted
- Unmodified

[Place an Order Now](#)

© Copyright 2010 Integrated DNA Technologies, Inc

# Place Order NOW!

**XX IDT**  
INTEGRATED DNA  
TECHNOLOGIES

Chat is now closed.  
Please click to email  
a representative.

[LogIn]  
Spain

0 Items € 0,00

Home Products Order Support Services SciTools Search Go

### Order Oligos

Change Form: 1 Expand to this many items  Duplex  Paste Go

25 nmole DNA Oligo = 15-60 bases    100 nmole DNA oligo = 10-90 bases    250 nmole DNA oligo = 5-100 bases  
1 μmole DNA oligo = 5-100 bases    5 μmole DNA oligo = 5-50 bases    10 μmole DNA oligo = 5-50 bases  
25 nmole Ultramer DNA Oligo = 60-200 bases    4 nmole Ultramer DNA Oligo = 60-200 bases    PAGE Ultramer DNA Oligo = 60-200 bases

Quantity:  Purification: Standard Desalting

Sequence Name:  # Bases: 21

5'-ACT GCA CCA TAA GCA ACT TTT-3'

Notes: Enter your notes here. Please do not enter modifications.

ADD TO ORDER  
ADD TO WISH LIST

Help 5' mods Internal Mods 3' mods Services Mixed Bases

**Preparative Services**  
 LabReady (more detail) € 2,82 EUR

**Customized Labels** (more detail)  
Stock IDT Label FREE



# DNA by Mail





# It runs!

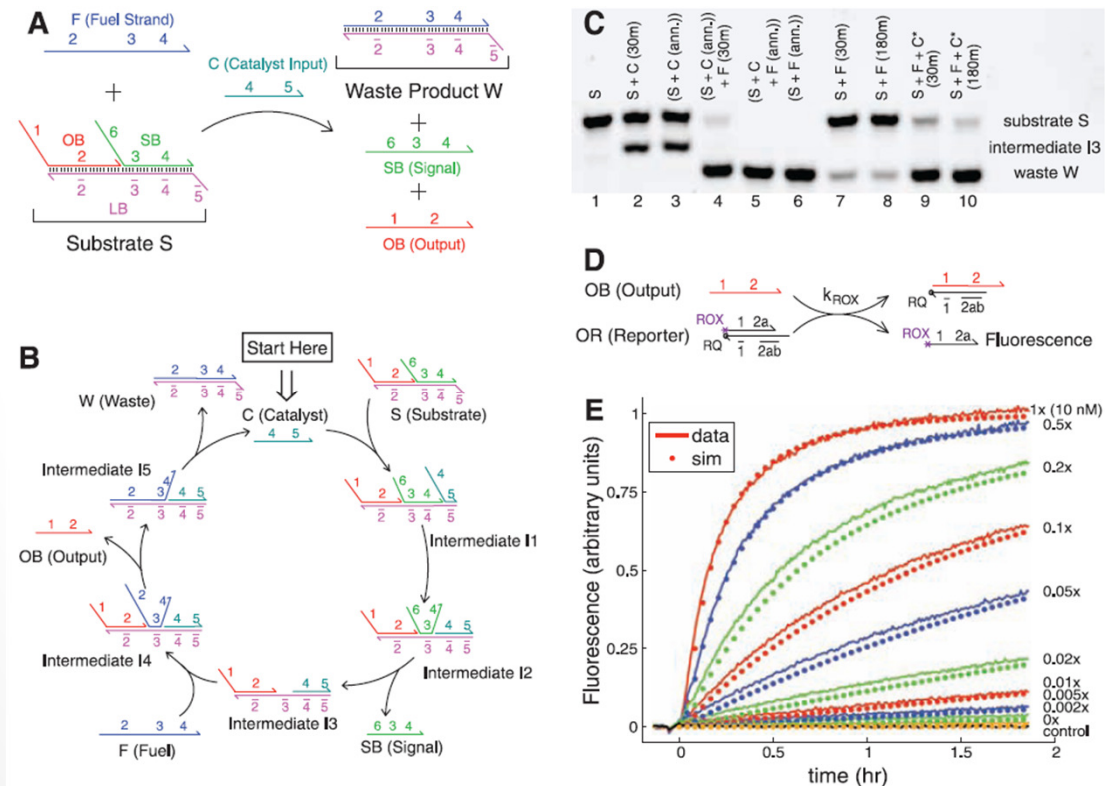
- (Nothing to do with us.)

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

David Yu Zhang, *et al.*

*Science* **318**, 1121 (2007);

DOI: 10.1126/science.1148532



# DNA Compilation

...

# Compilers

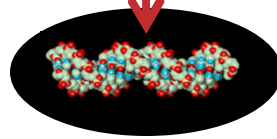
Monolithic  
Compilers



Language  
Design #1

Boolean  
Networks

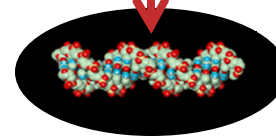
Language  
Implementation #1



Language  
Design #2

Petri  
Nets

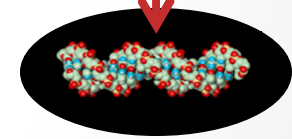
Language  
Implementation #2



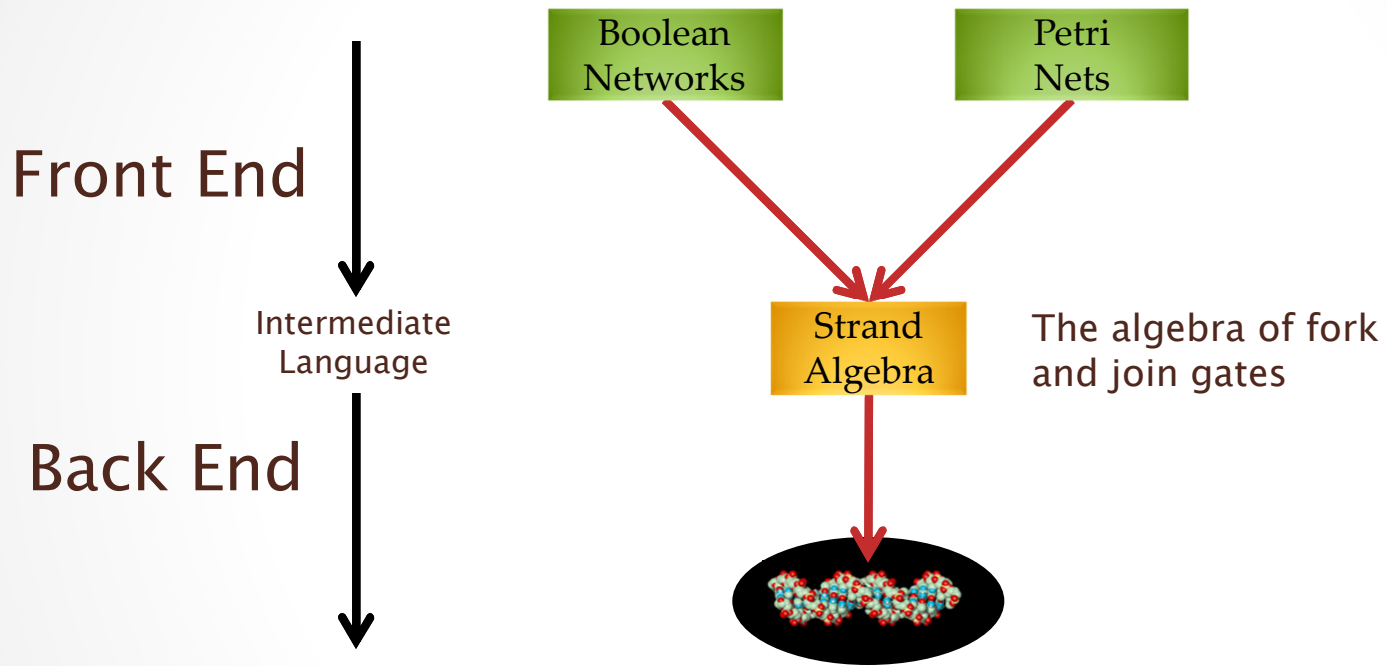
Language  
Design #3

...

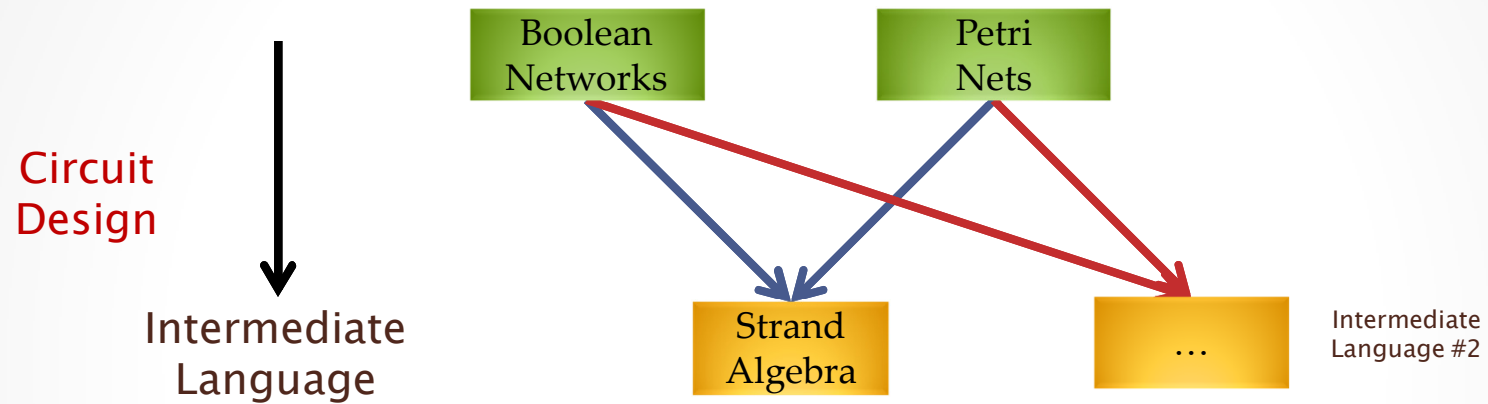
Language  
Implementation #3



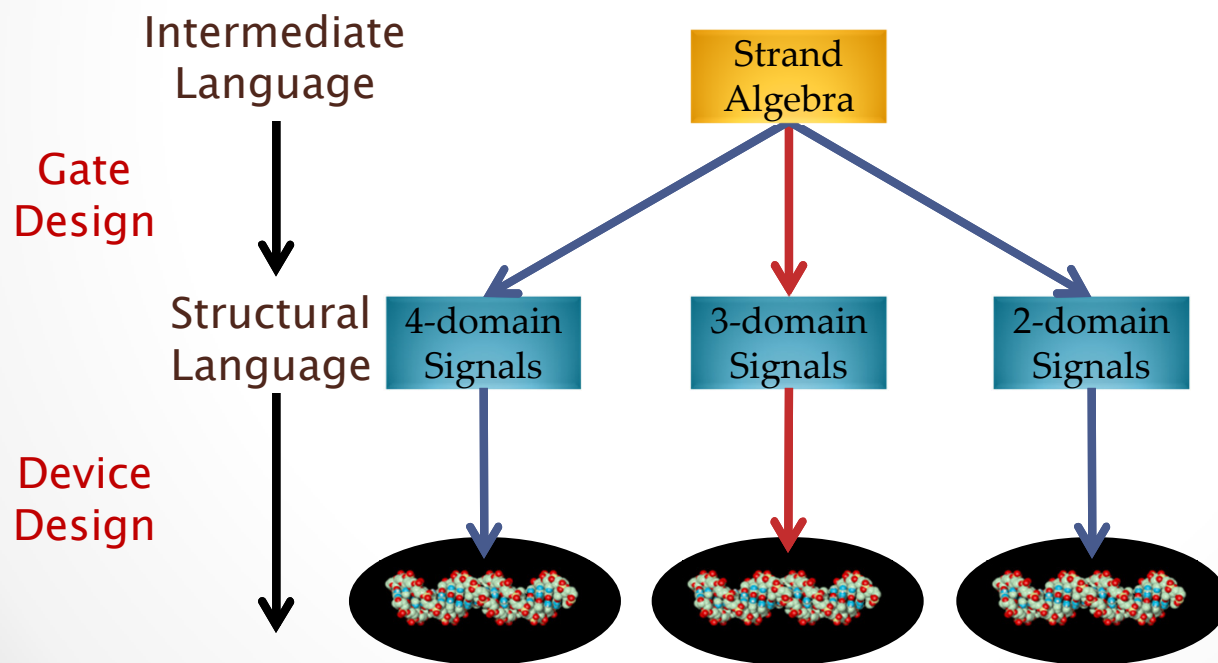
# Intermediate Languages



# Front Ends



# Back Ends



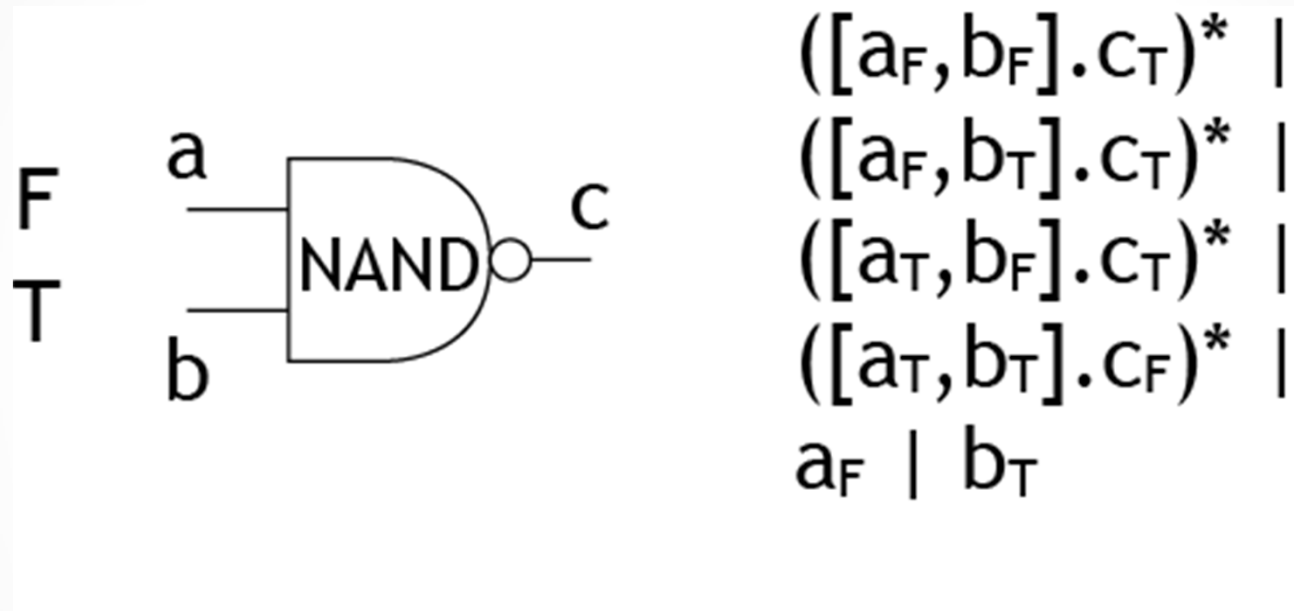
# Compiling Abstract Machines

...



# Boolean Networks

## Boolean Networks to Strand Algebra

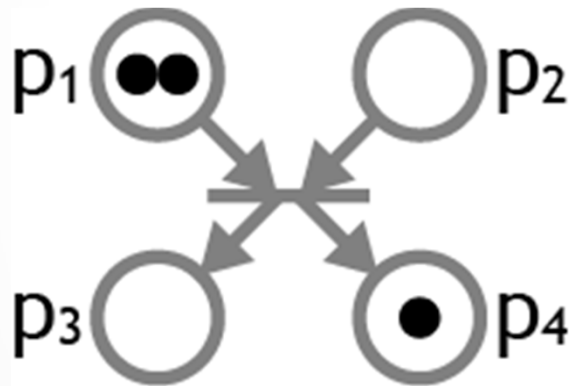


- This encoding is *compositional*, and can encode *any* Boolean network:
- multi-stage networks can be assembled (**combinatorial logic**)
  - network loops are allowed (**sequential logic**)

# Petri Nets

## Petri Nets to Strand Algebra

Transitions as Gates  
Place markings as Signals



$$([p_1, p_2] \cdot [p_3, p_4])^* \mid p_1 \mid p_1 \mid p_4$$

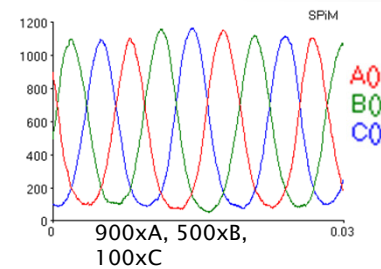
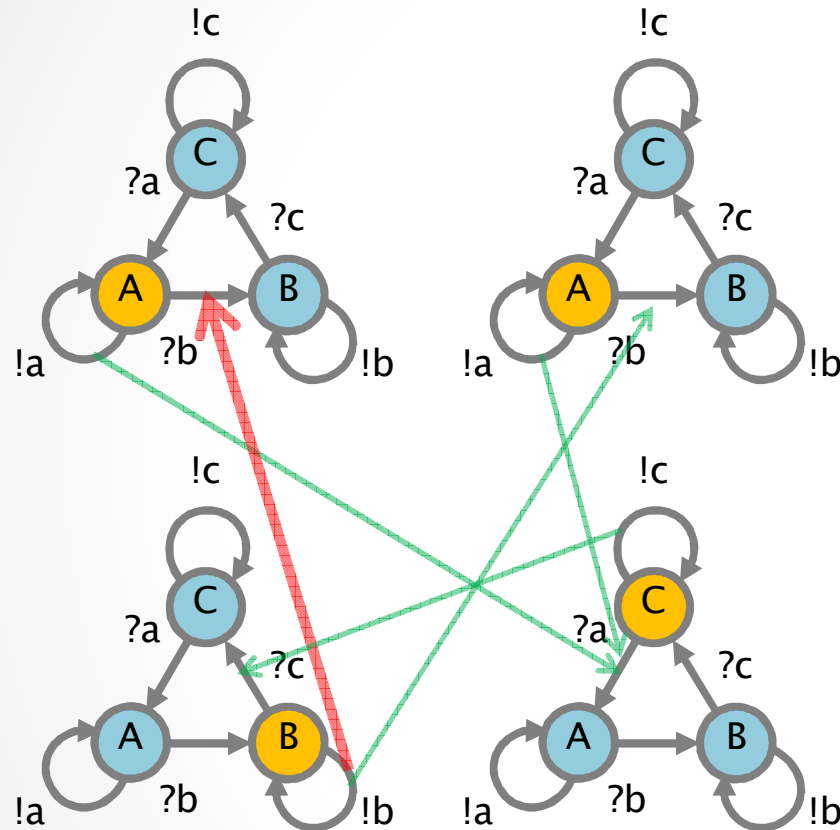
# Chemical Reaction Networks

Implementing an arbitrary finite chemical system in  
DNA with asymptotically correct kinetics  
Soloveichik & al. DNA 15

Species become signals  
Reactions become gates



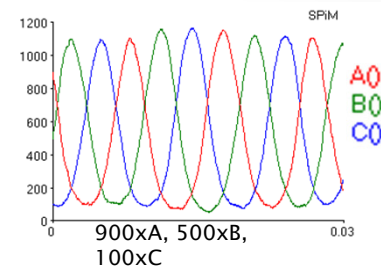
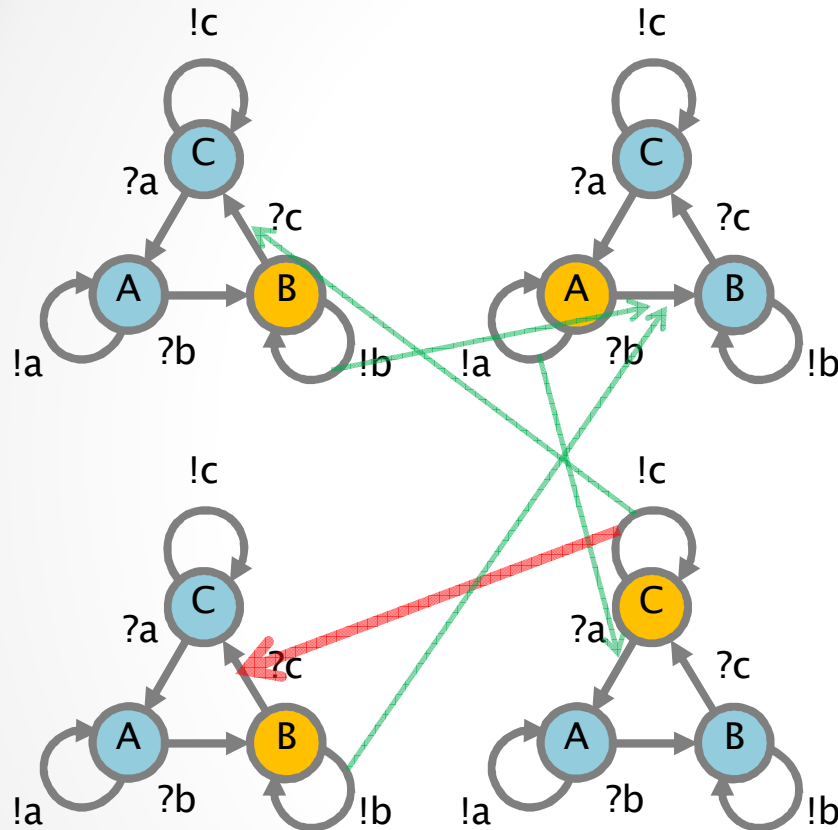
# Interacting Automata



$([A, B]. [B, B])^* \mid$   
 $([B, C]. [C, C])^* \mid$   
 $([C, A]. [A, A])^* \mid$   
 $A \mid A \mid B \mid C$

This is a uniform population of identical automata,  
 but heterogeneous populations of interacting automata can be similarly handled.

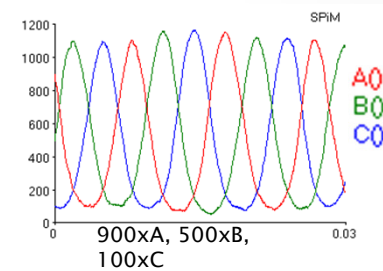
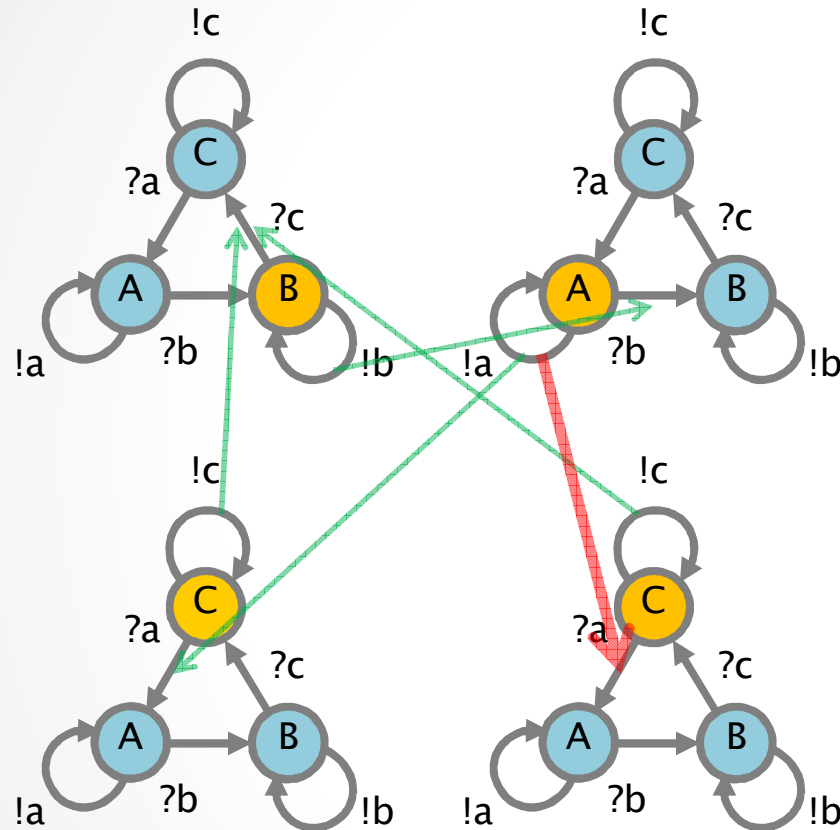
# Interacting Automata



$([A, B]. [B, B])^* \mid$   
 $([B, C]. [C, C])^* \mid$   
 $([C, A]. [A, A])^* \mid$   
 $A \mid B \mid B \mid C$

This is a uniform population of identical automata,  
 but heterogeneous populations of interacting automata can be similarly handled.

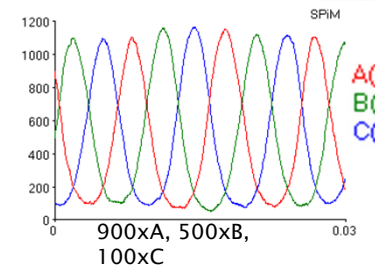
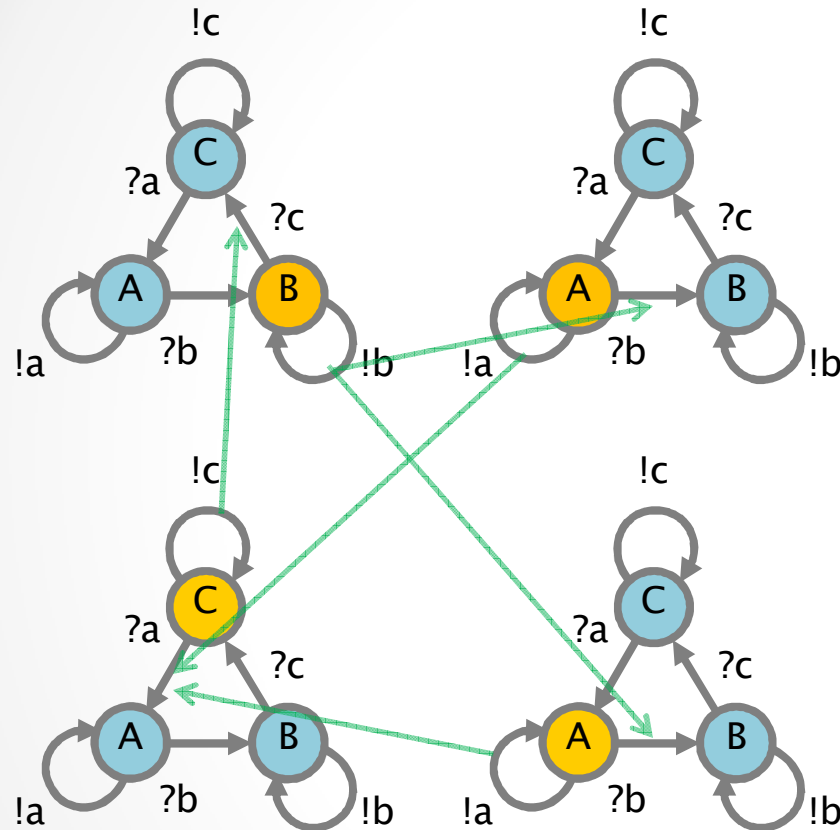
# Interacting Automata



$([A, B]. [B, B])^* \mid$   
 $([B, C]. [C, C])^* \mid$   
 $([C, A]. [A, A])^* \mid$   
 $A \mid B \mid C \mid C$

This is a uniform population of identical automata,  
 but heterogeneous populations of interacting automata can be similarly handled.

# Interacting Automata



$([A, B]. [B, B])^* \mid$   
 $([B, C]. [C, C])^* \mid$   
 $([C, A]. [A, A])^* \mid$   
**A** **A** **B** **C**

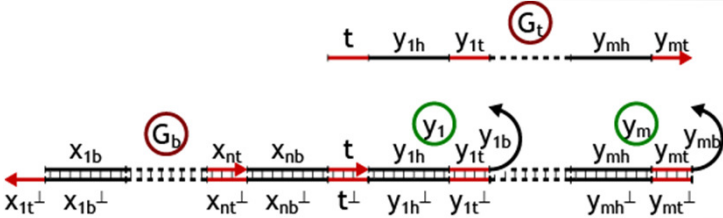
This is a uniform population of identical automata,  
 but heterogeneous populations of interacting automata can be similarly handled.



# Strand Algebra to DSD

$P ::= x \mid [x_1, \dots, x_n]. [y_1, \dots, y_m] \mid 0 \mid P \mid P \mid P^* \quad n \geq 1, m \geq 0$

•  $\text{compile}(x) =$  

•  $\text{compile}([x_1, \dots, x_n]. [y_1, \dots, y_m]) =$  

•  $\text{compile}(0) =$  empty solution

•  $\text{compile}(P \mid P') = \text{mix}(\text{compile}(P), \text{compile}(P'))$

•  $\text{compile}(P^*) = \text{population}(\text{compile}(P))$

**And finally...**

...

# Summary

- Abstract Machines to Strand Algebra
  - Or other intermediate language
- Strand Algebra to DSD
  - Or other structural language
- Simulation, analysis, etc.
  - Iterate a lot
- DSD to Sequences
  - E.g. NuPack, or pre-build strand libraries
- Sequences to DNA
  - Web order
- DNA experiments
  - Fairly basic wet lab
- Deployable Nanotech

# Conclusions

- **Nucleic Acids**
  - Programmable matter
- **DNA Strand Displacement**
  - A computational mechanism at the molecular level
- **DNA as a Compilation Target for Abstract Machines**
  - Abstract Machines (Boolean Networks, Petri Nets, Interacting Automata)
  - Intermediate languages (Strand Algebra, Strand Displacement Language).
  - DNA sequence generation.
- **Tools**
  - Thermodynamic analysis.
  - Reaction graph generation.
  - Simulation.
  - Verification (not yet).

# Acknowledgments



- Illustrations
  - John Reif, Duke
  - Ned Seeman, NYU
  - Erik Winfree, Caltech
  - Bernard Yurke, Boise State
  - Wikipedia
  - YouTube
- David Soloveichik

