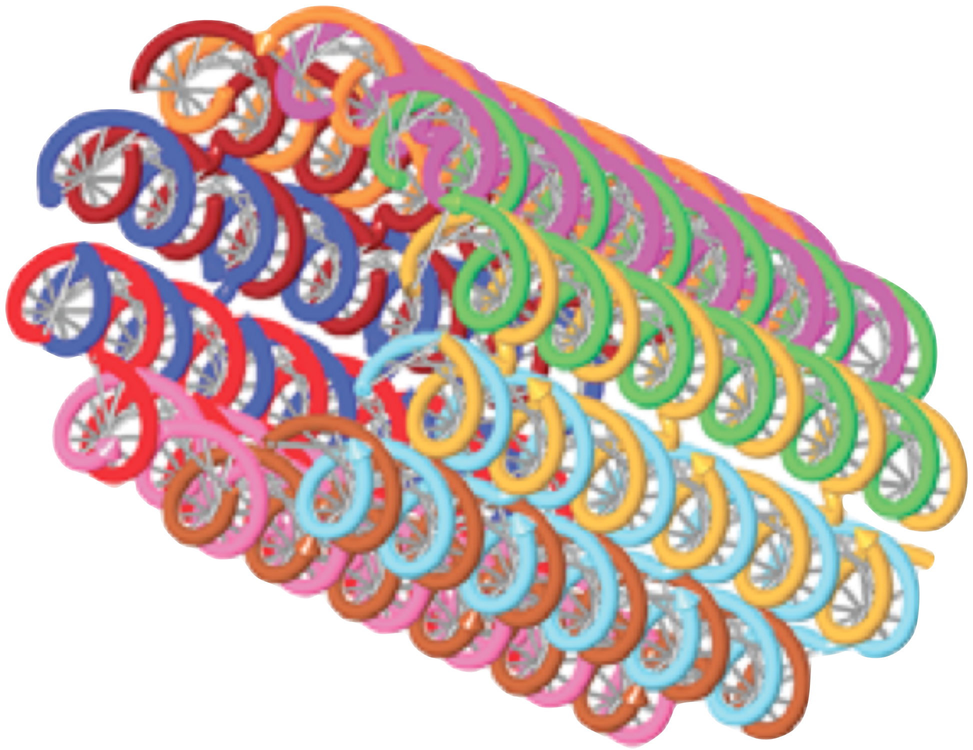
An experimental realization of a universal computer

Nicolas Schabanel
CNRS
LIP & IXXI - ÉNS de Lyon

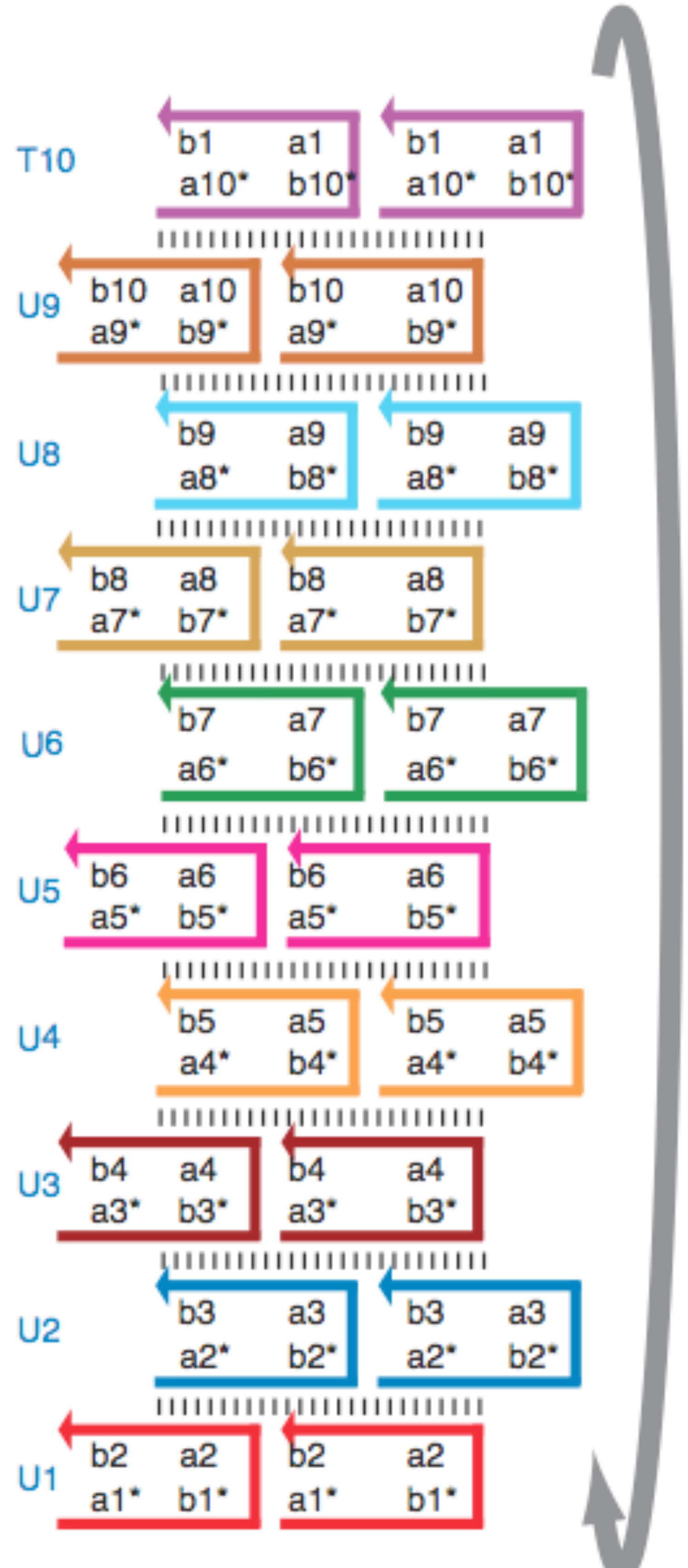
Slides mainly borrowed from Damien Woods et al (Nature 2019)

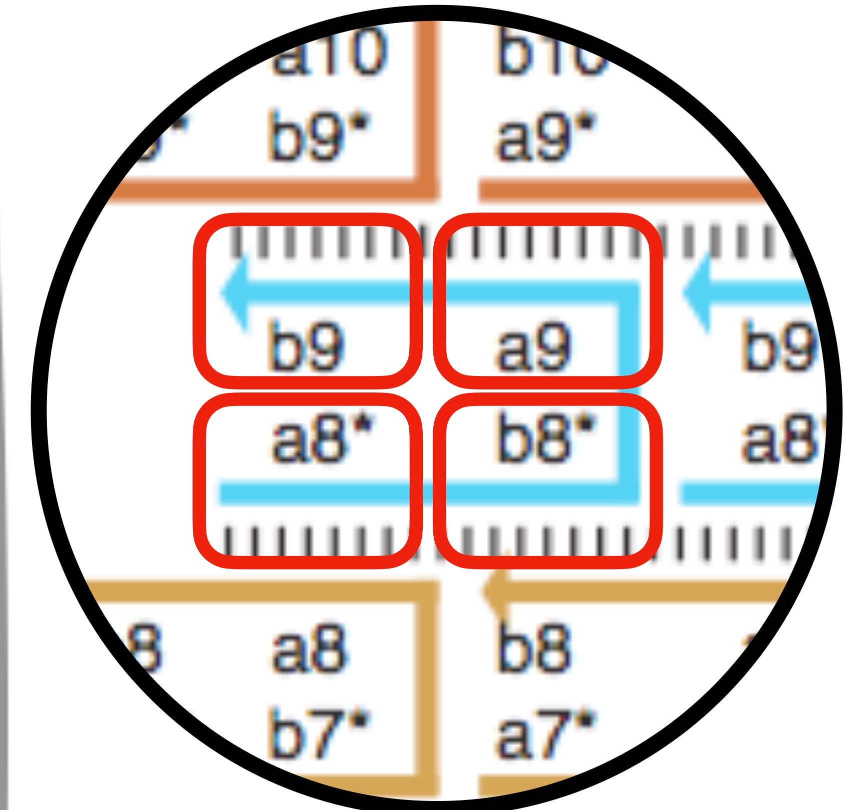
Single Stranded Tiles Nanotubes

Single stranded Nanotubes

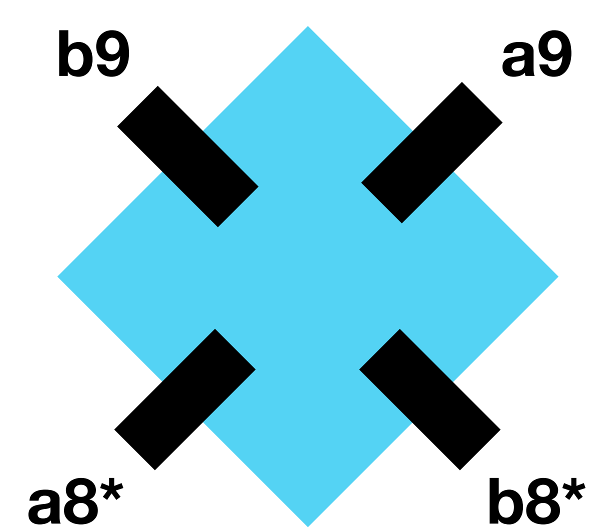


10-helix nanotube schematic,
Yin et al. ’08

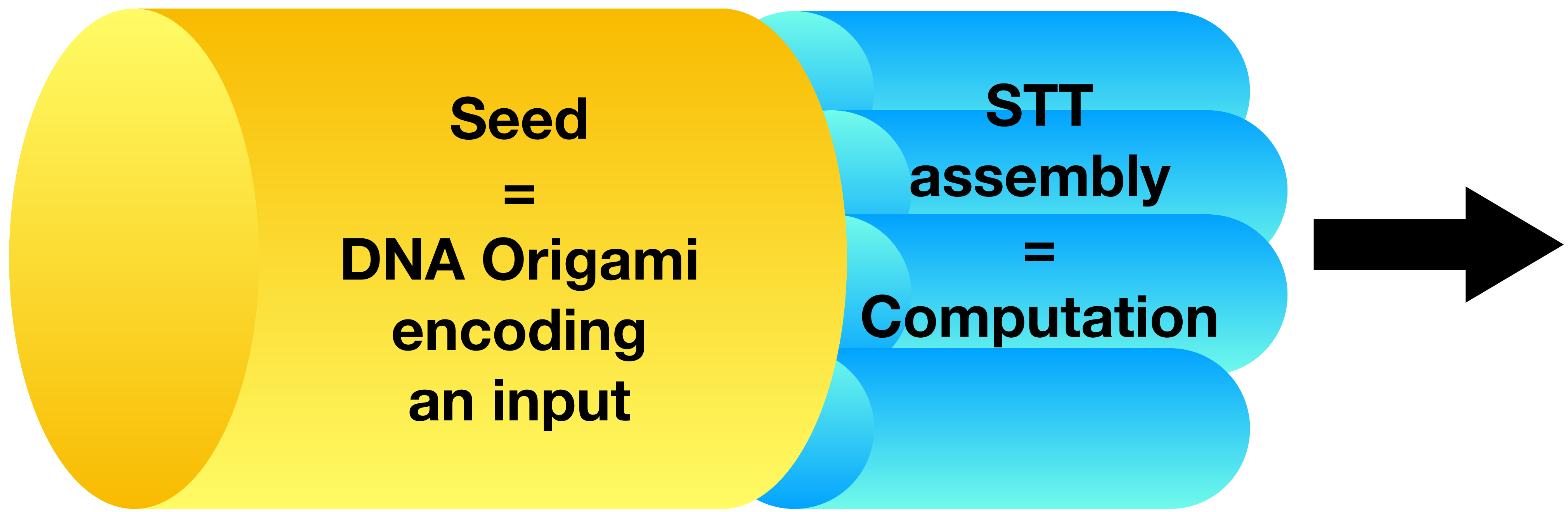


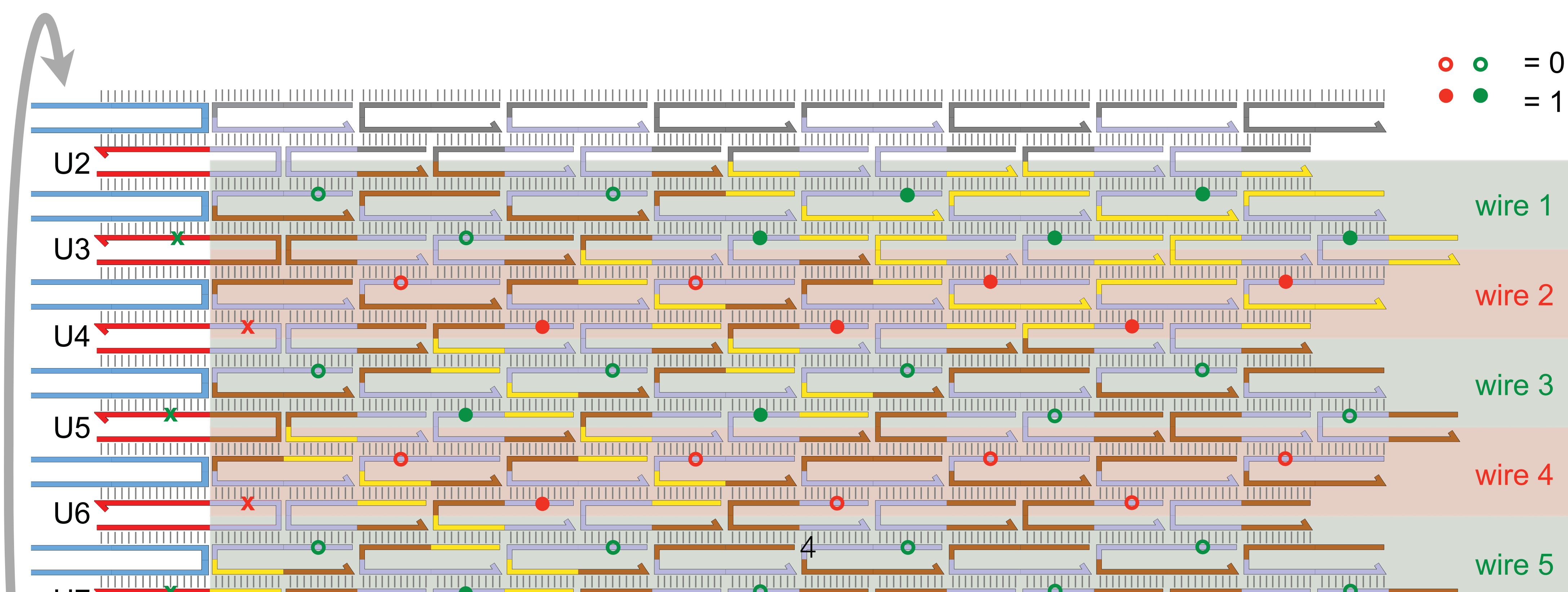


4 domains = 4 glues

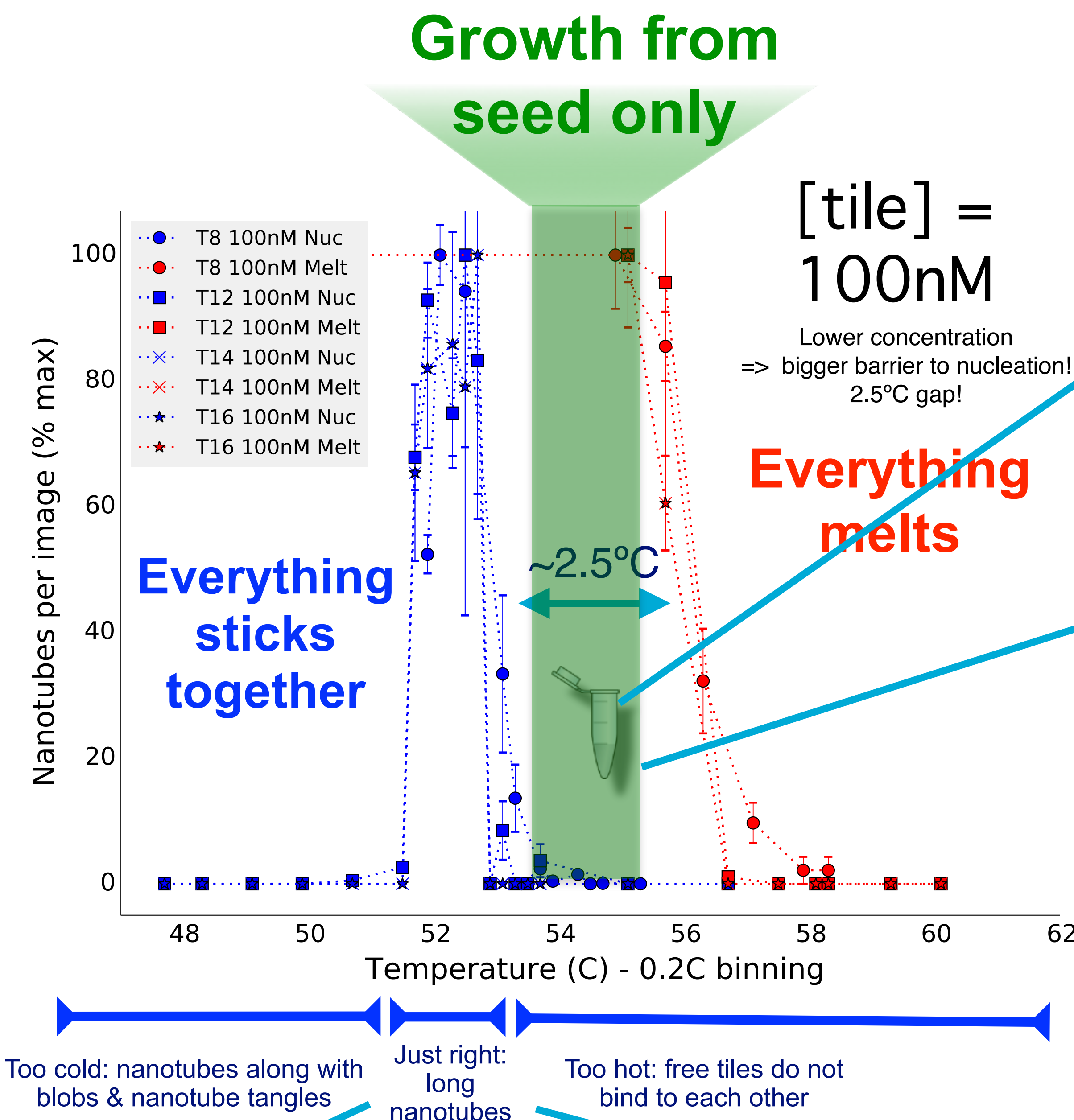


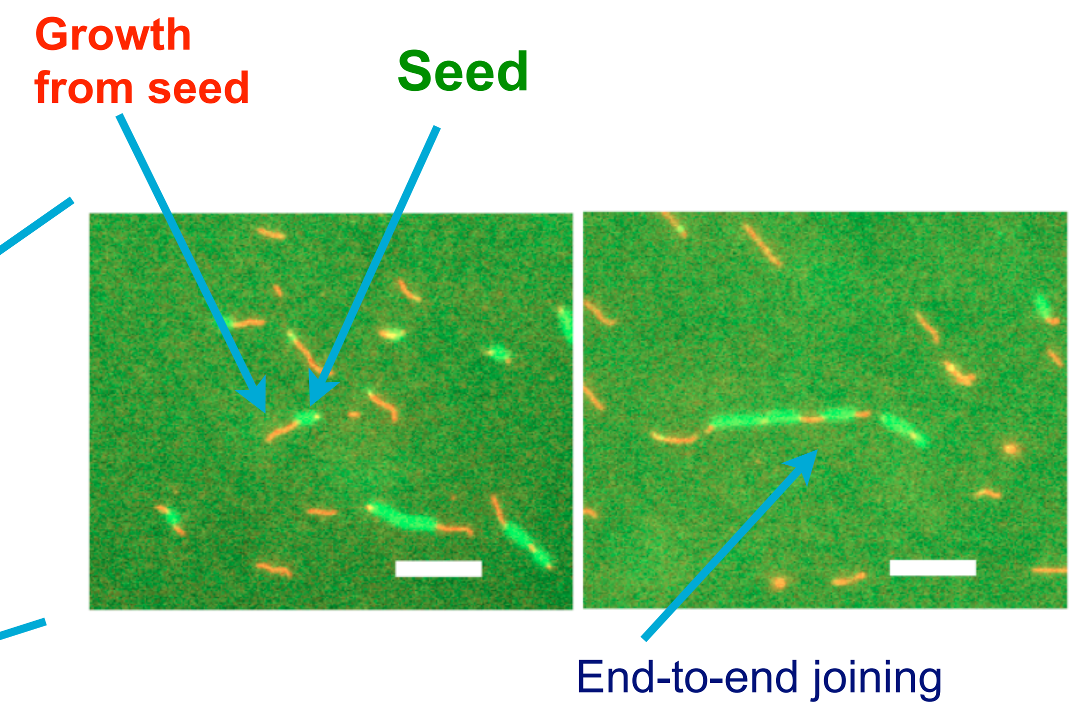
Growing them





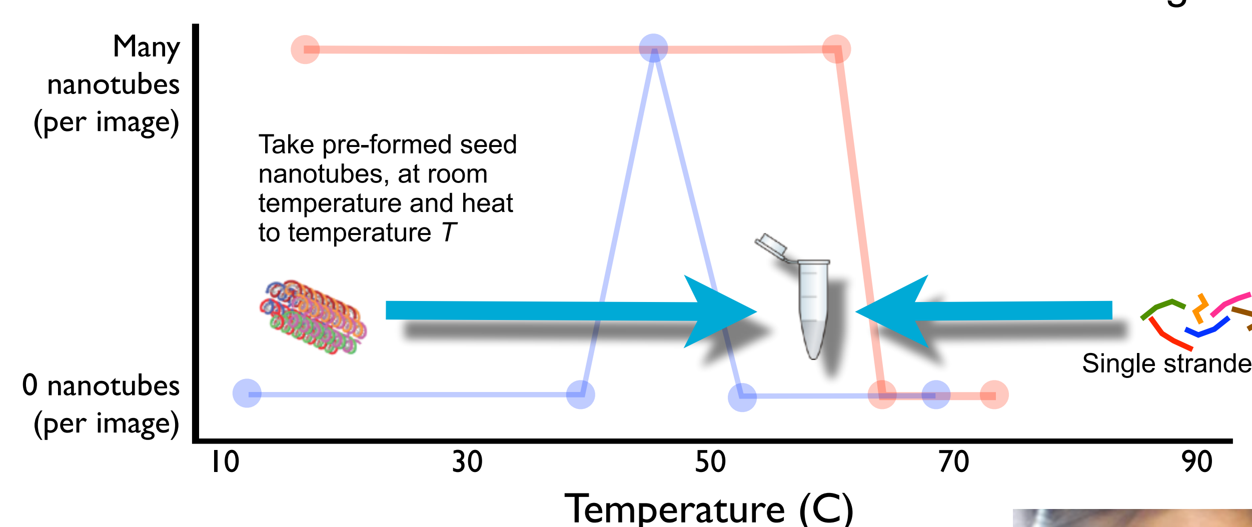
Seeded growth: barrier to nucleation at [tile]=100nM



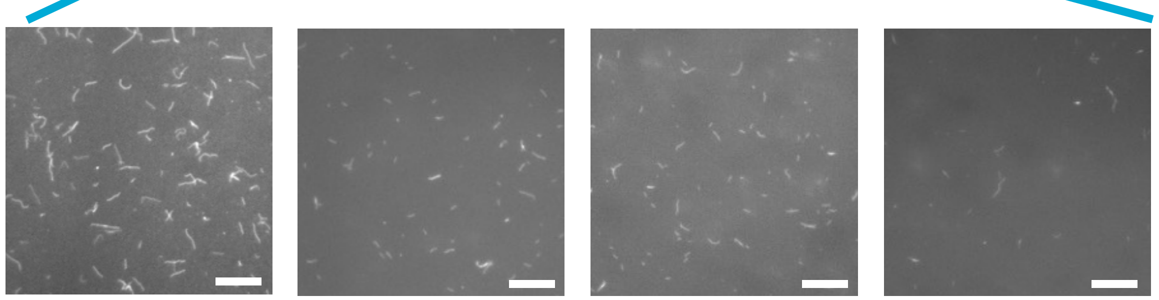


seeds: 8-helix nanotubes, Alexa647 labelled
growth from seed: 8-helix tiles, Cy3 labelled
scale bar: 10μm. 100nM tile concentration

Controls: *0* seed nanotubes => 0 nanotubes/image

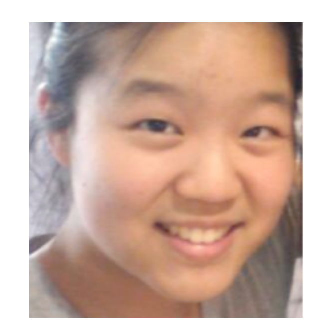


We work at this temperature



8-helix. 100nM. 53.0 C — 12-helix. 100nM. 52.2 C — 14-helix. 100nM. 52.2 C — 16-helix. 100nM. 52.2 C

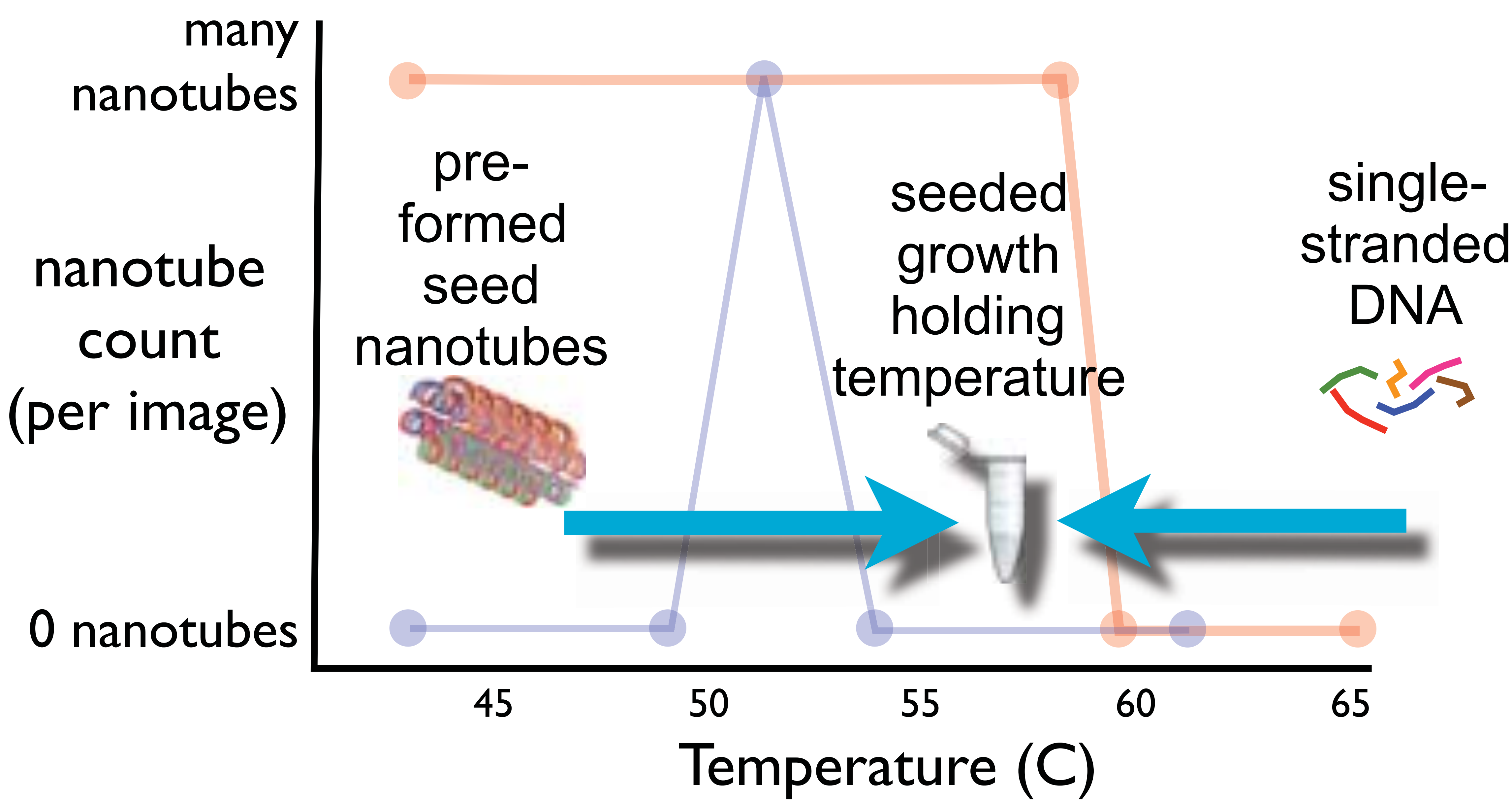
Scale bars 10μm. ~24 hour temperature hold experiments. cy3 label



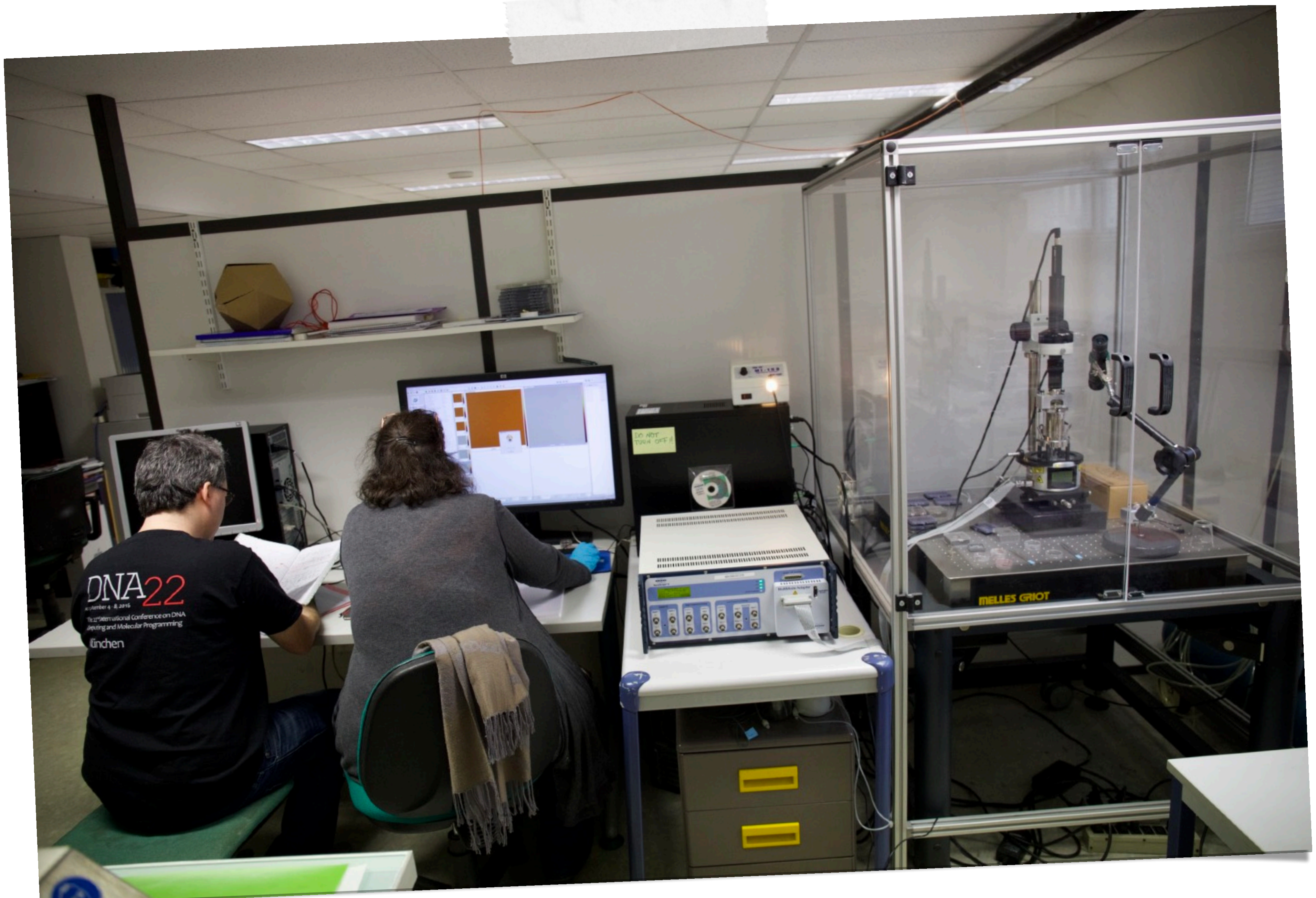
Joy Hui

Damien Woods

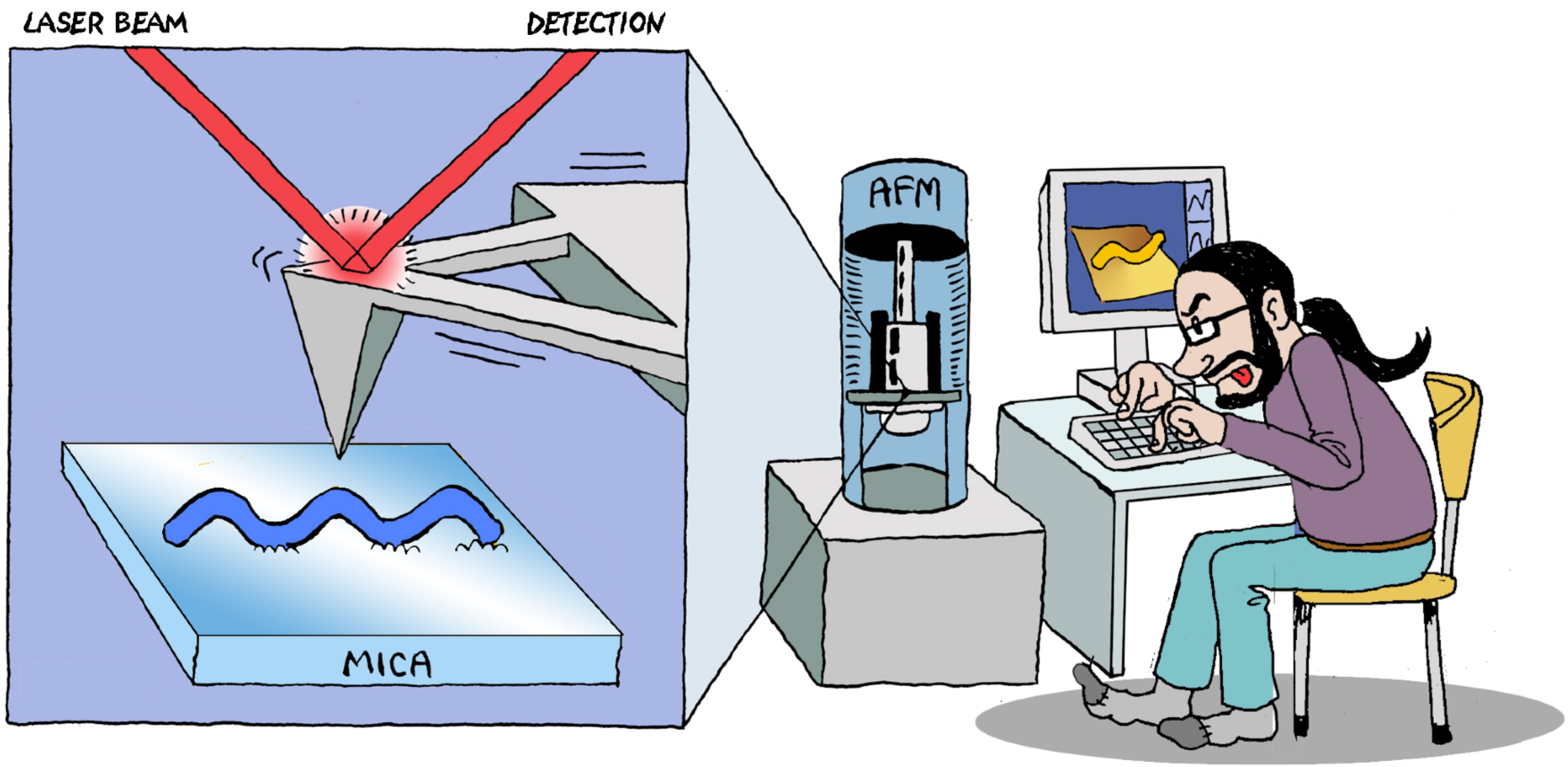
Seeded growth only



Imaging the results



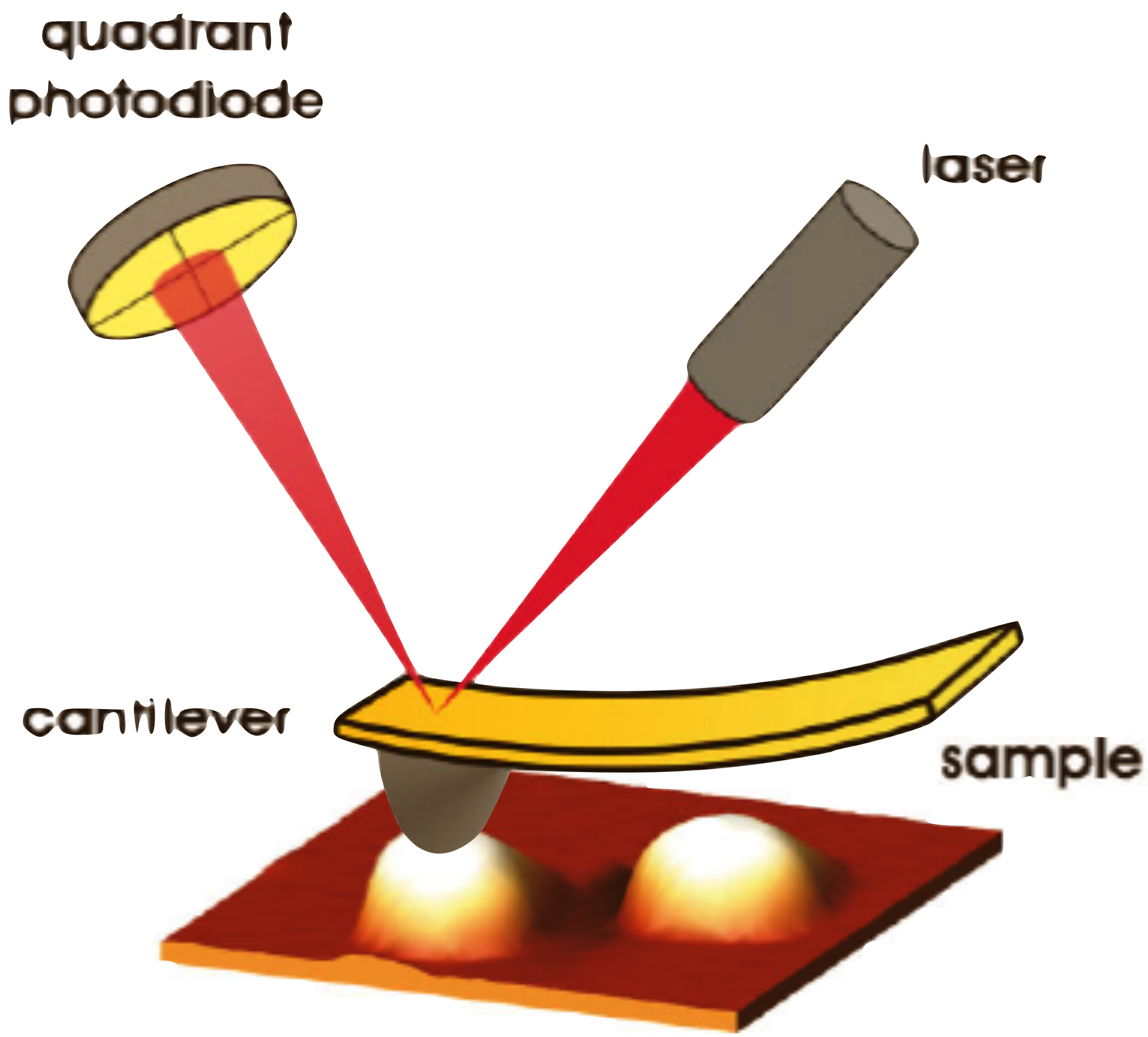
Principle of Atomic Force Microscopy

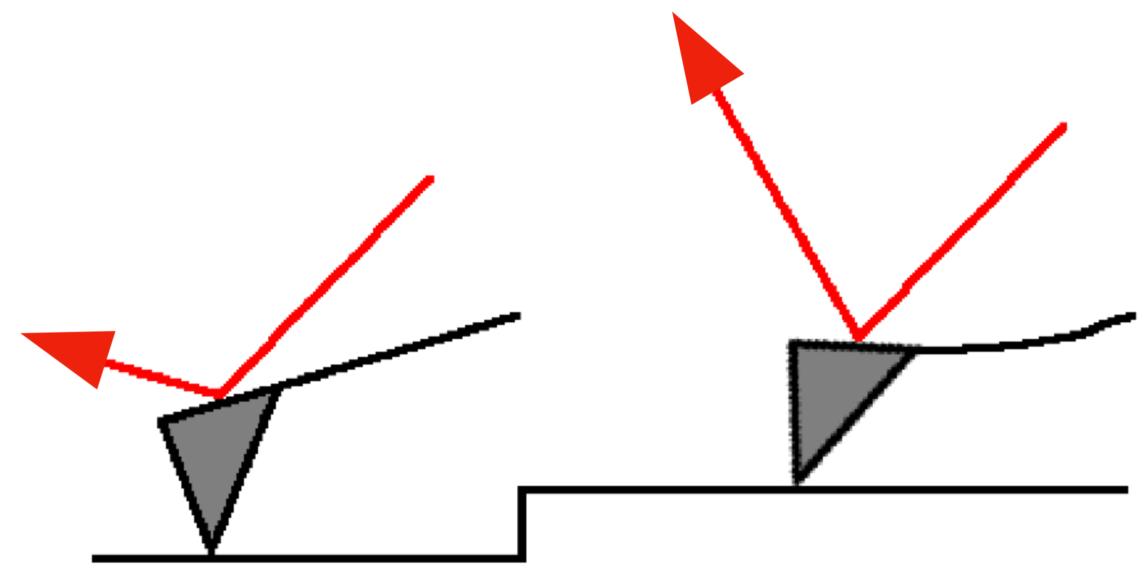


The microscope works by scanning the surface with a sharp probe and gently touching the DNAs that arrange on the mica.

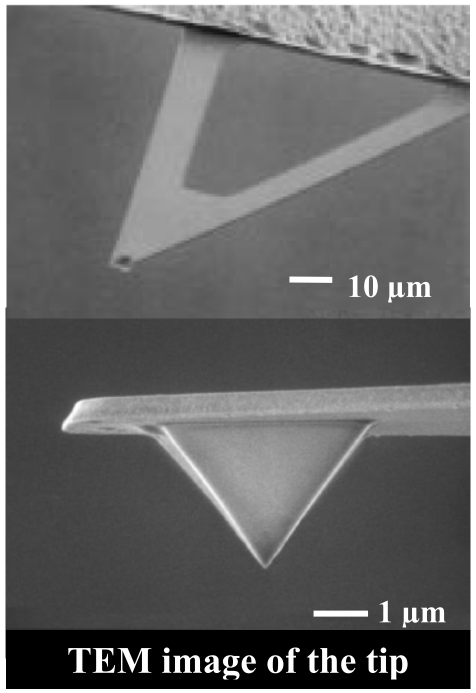
(Artwork: Ebbe Andersen- Slide by Cody Geary)

Laser deflection

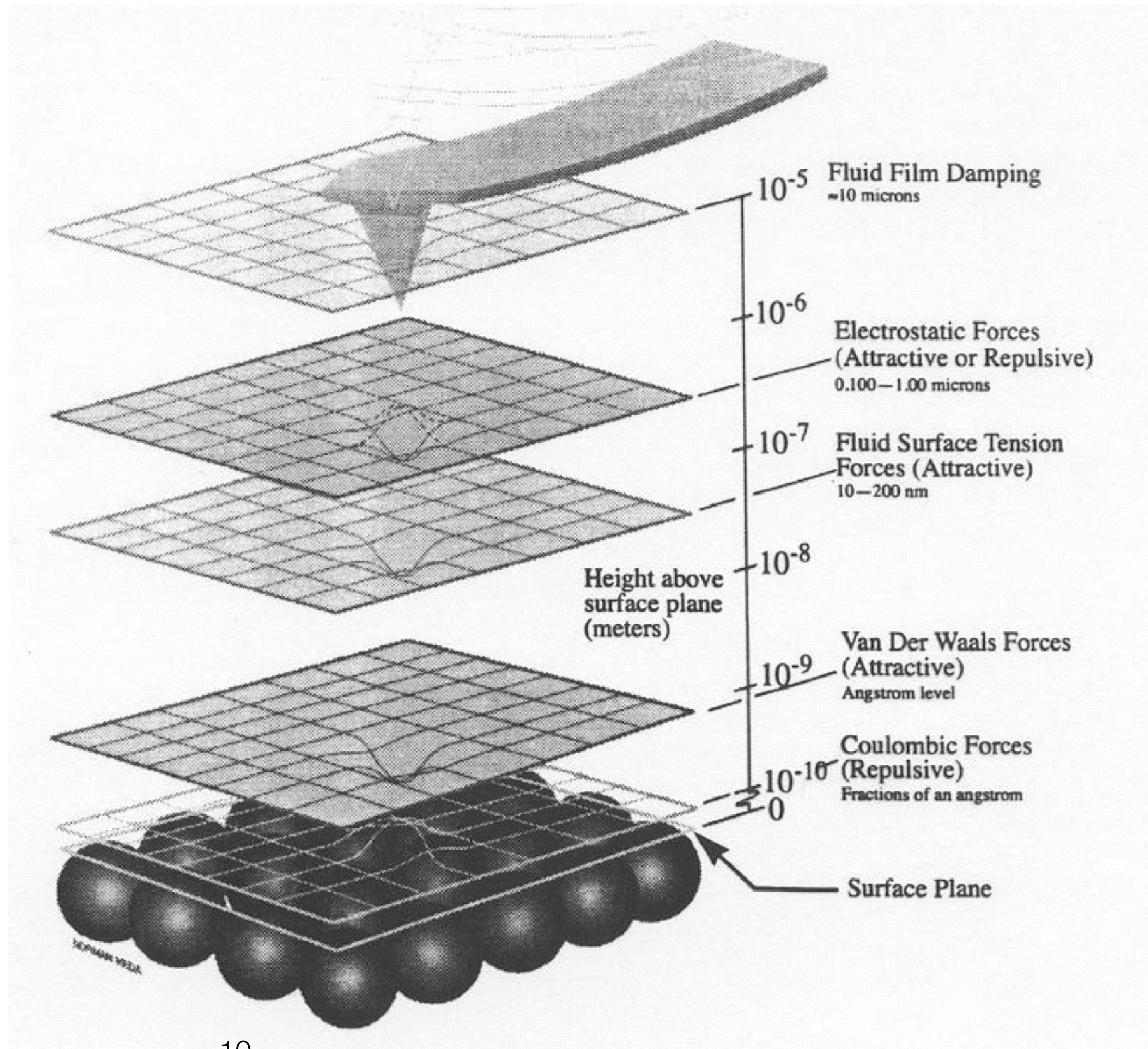




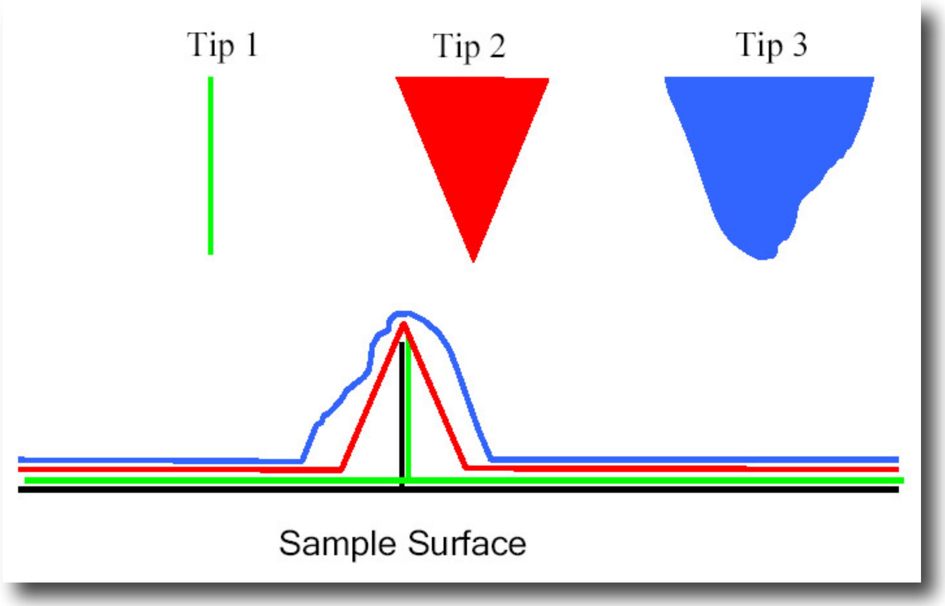
The forces involved in AFM

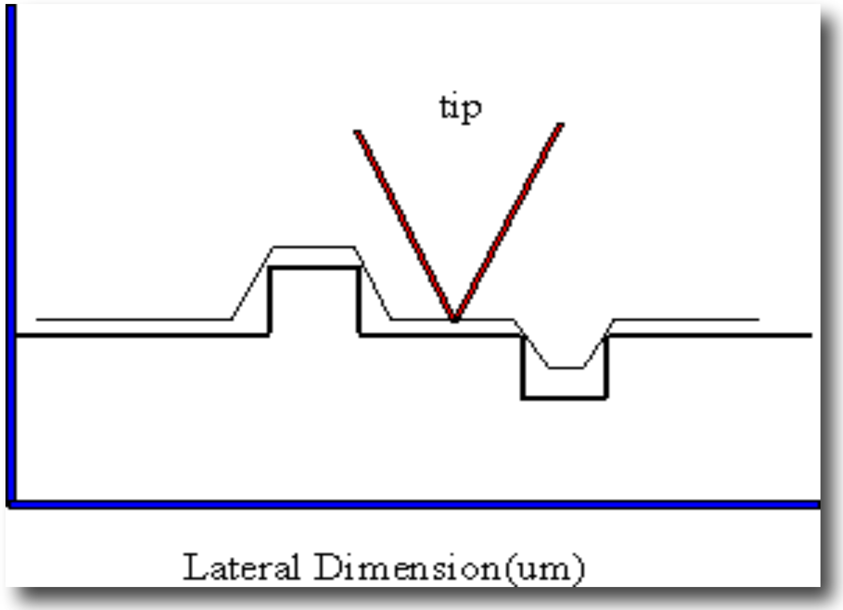


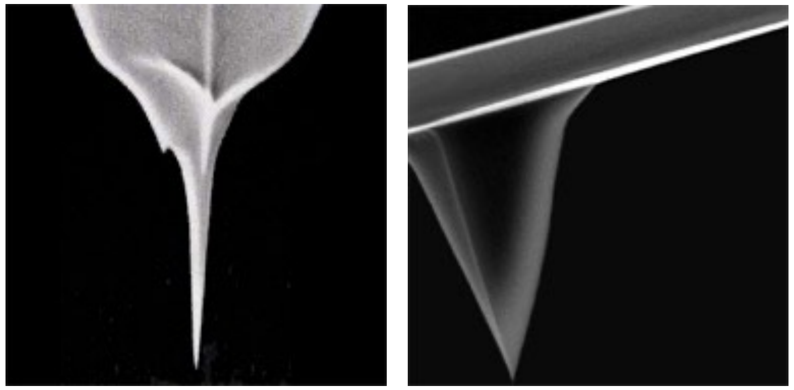
They are interaction forces between the atoms of the end of the tip and the atoms on the sample surface.



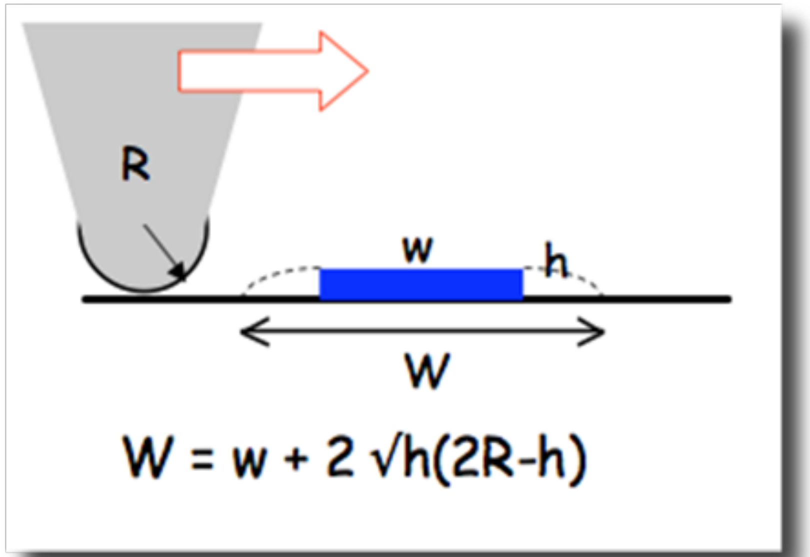
Tip convolution

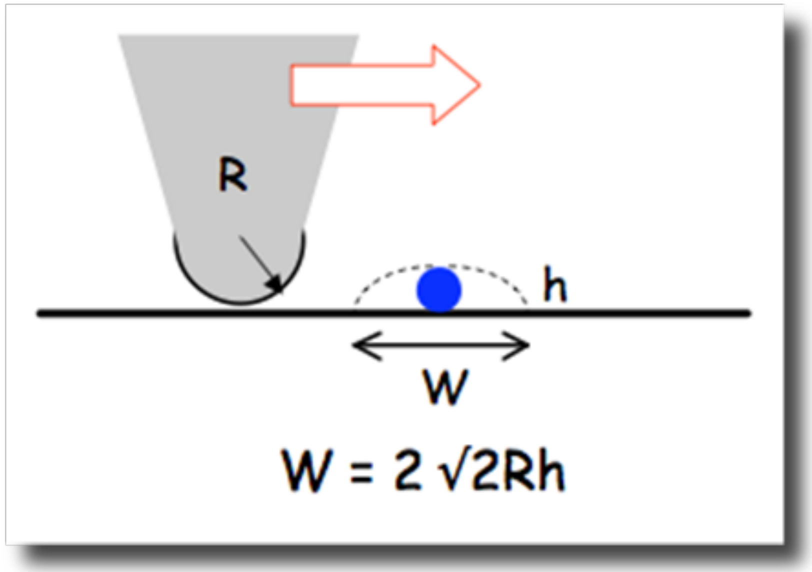






Tip radius 2-20 nm





Moskalenko

High resolution imaging

The Chemical Structure of a Molecule Resolved by Atomic Force Microscopy
Leo Gross, *et al.*
Science **325**, 1110 (2009);
DOI: 10.1126/science.1176210

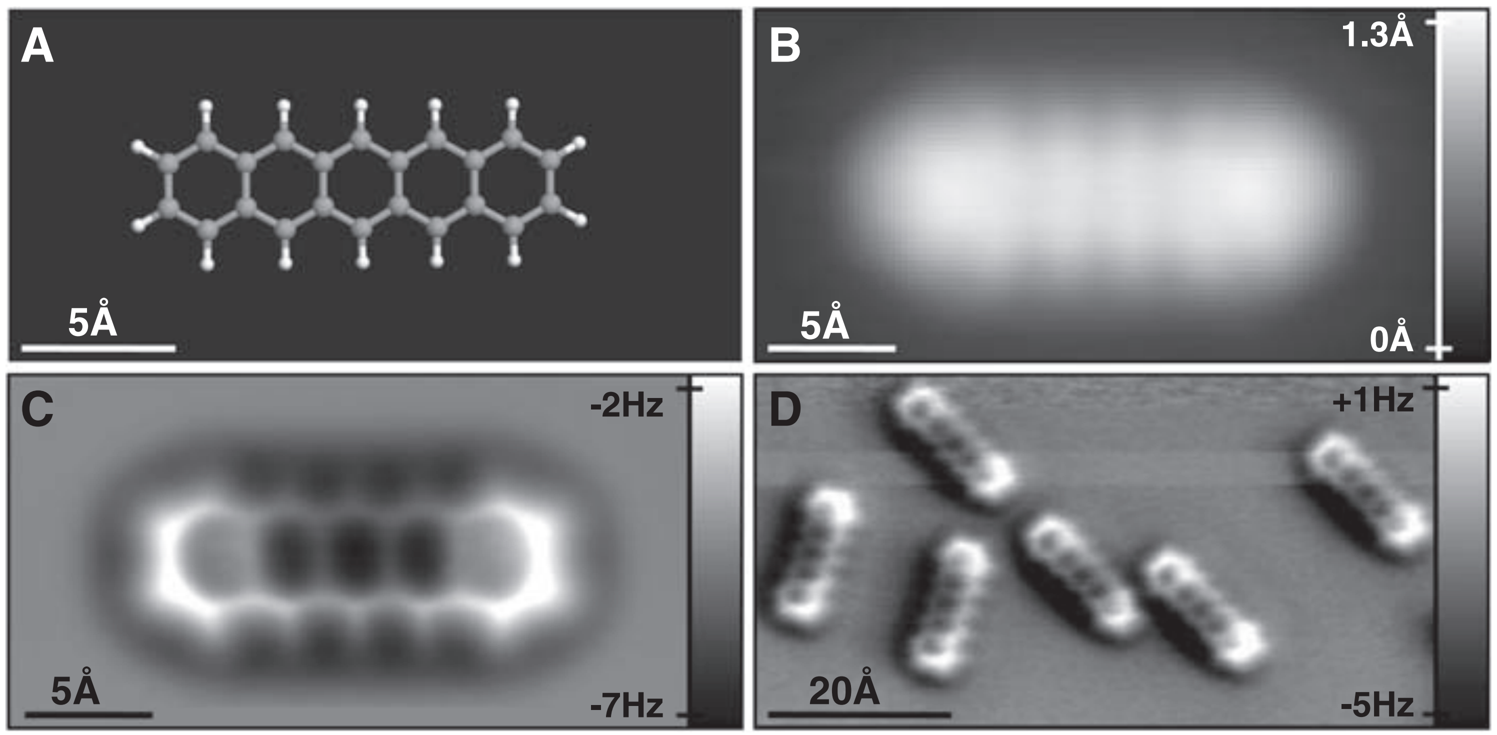
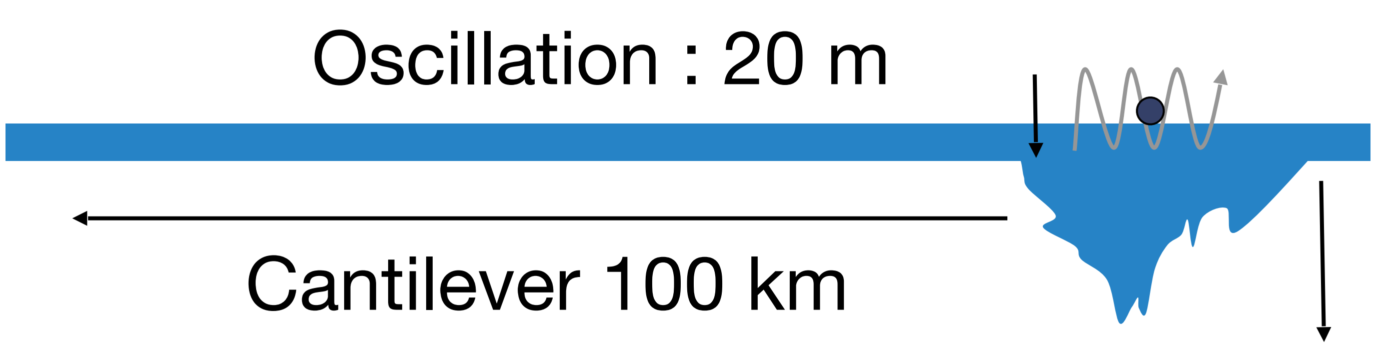
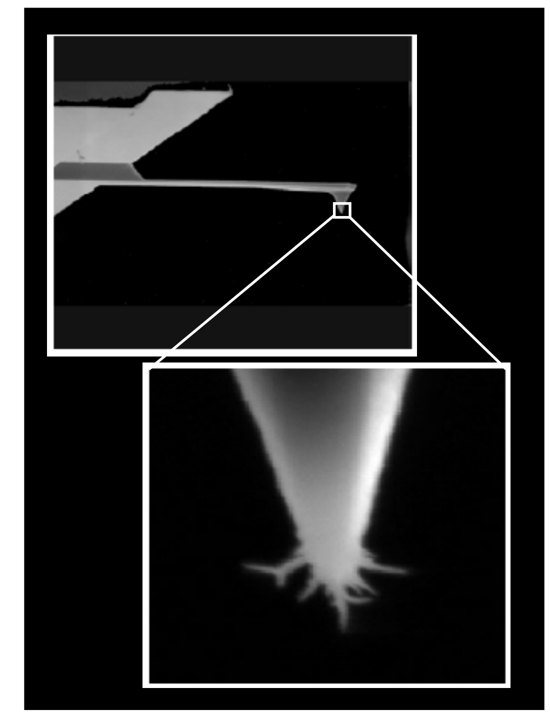


Fig. 1. STM and AFM imaging of pentacene on Cu(111). (**A**) Ball-and-stick model of the pentacene molecule. (**B**) Constant-current STM and (**C** and **D**) constant-height AFM images of pentacene acquired with a CO-modified tip. Imaging parameters are as follows: (B) set point I = 110 pA, V = 170 mV; (C) tip height z = –0.1 Å [with respect to the STM set point above Cu(111)], oscillation amplitude A = 0.2 Å; and (D) z = 0.0 Å, A = 0.8 Å. The asymmetry in the molecular imaging in (D) (showing a "shadow" only on the left side of the molecules) is probably caused by asymmetric adsorption geometry of the CO molecule at the tip apex.

About AFM scale

… how to shake the Mont Blanc over little men heads without crushing them

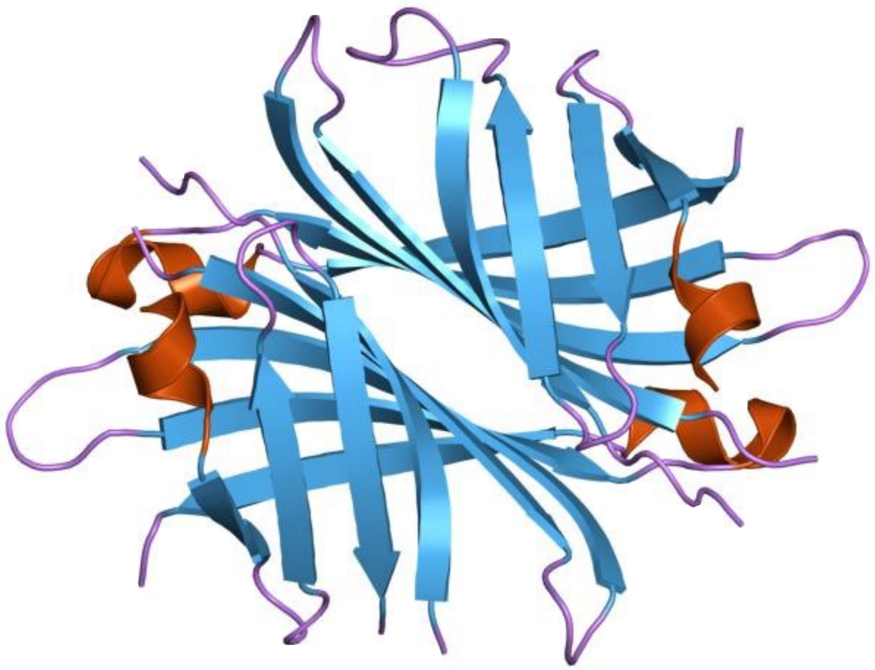




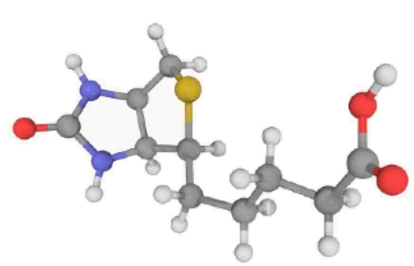
expanding to our scale : multiply all by 10^9

Marking 0s and 1s

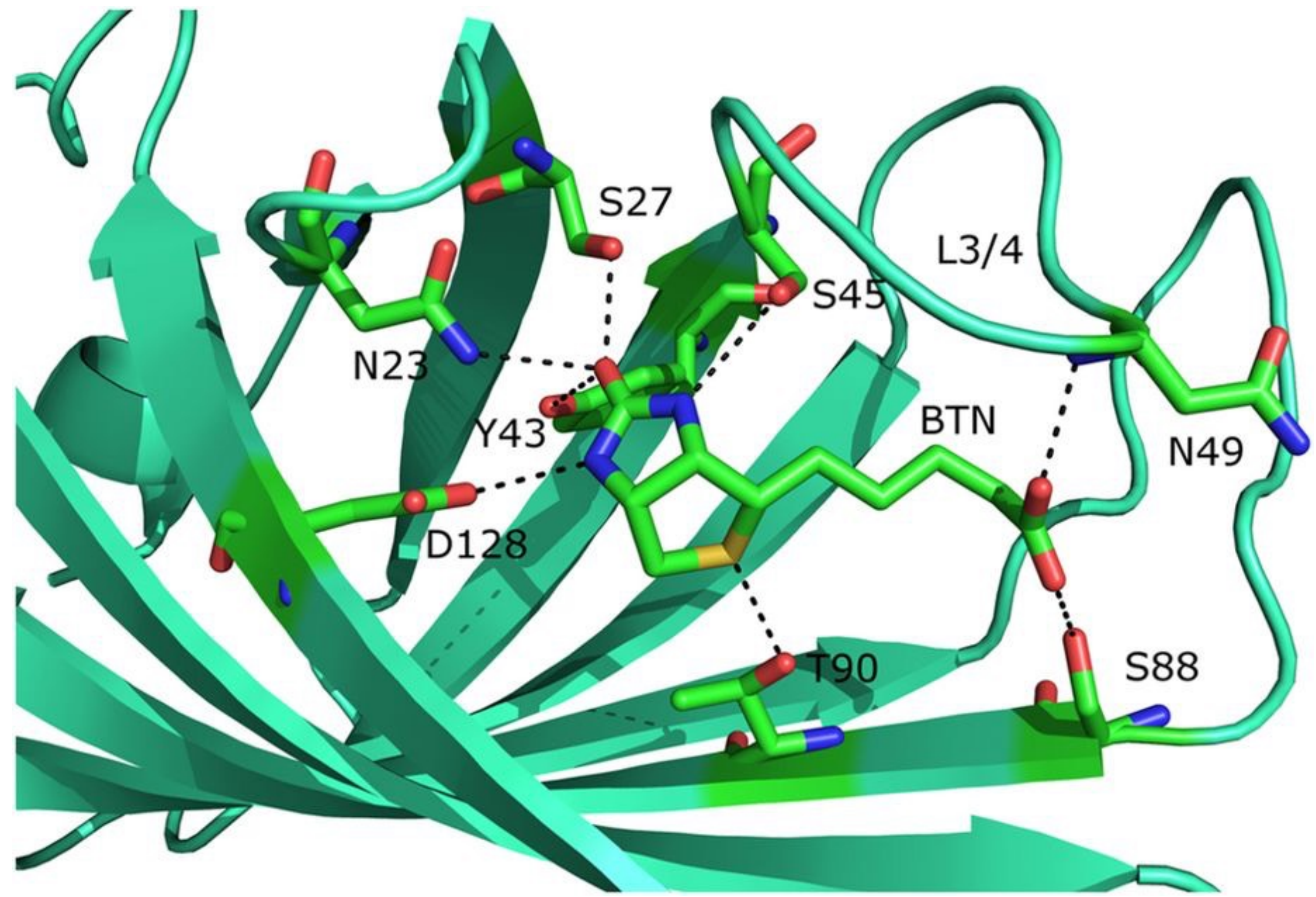
Streptavidin-biotin marker



Streptavidin : a "huge blob"



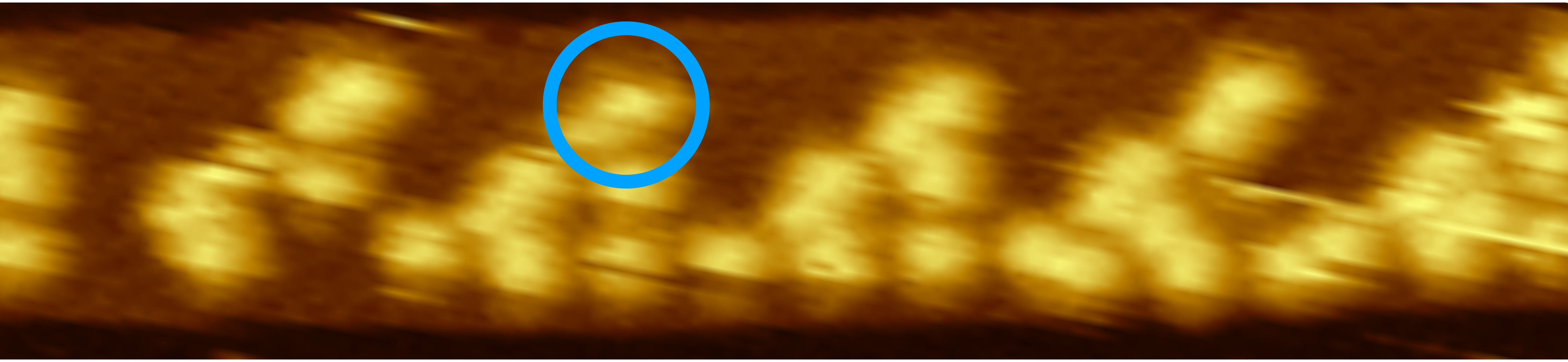
Biotin can easily be attached to DNA strand at order



Together they make one of the <u>strongest </u>non-covalent bond

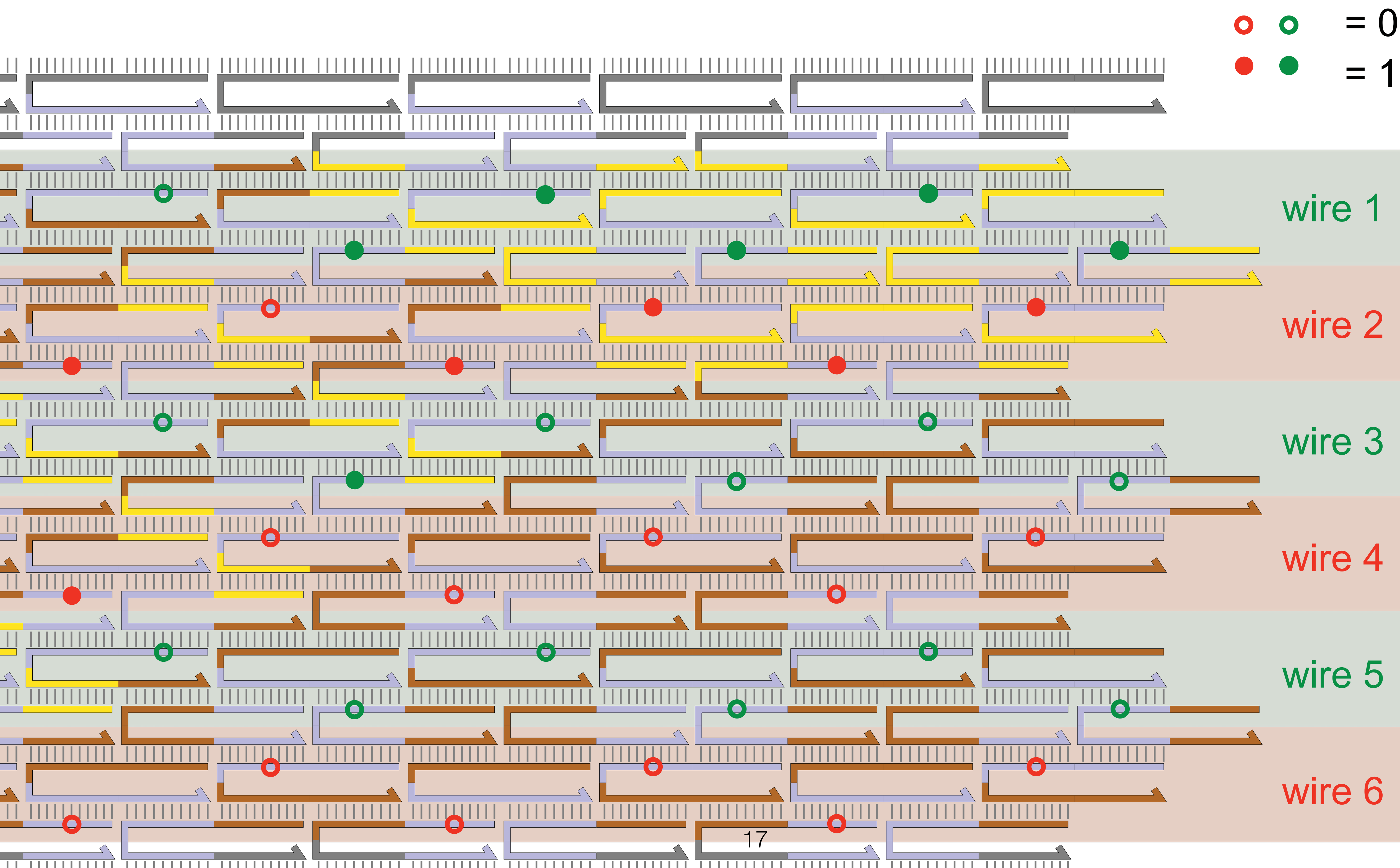
Streptavidin-biotin marks

We can order single DNA strand with biotin attached
(the tiles encoding a 1!)



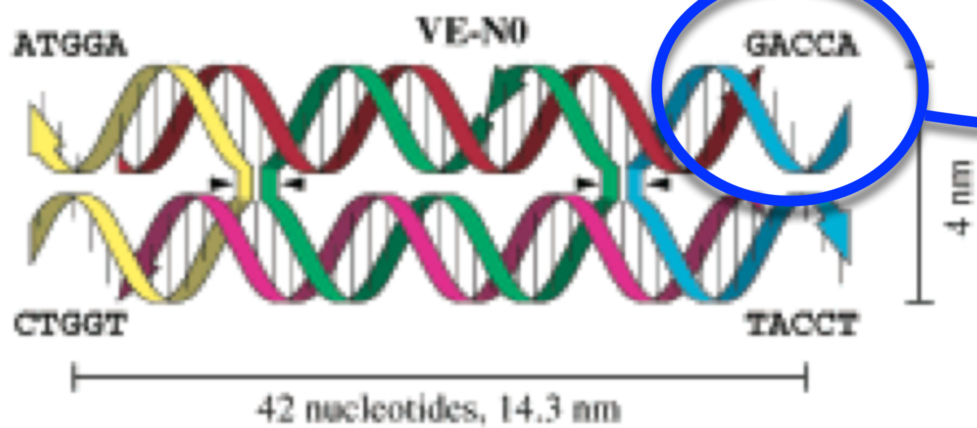
When added to the solution while imaging,
Streptavidin attaches to biotin,
marking the corresponding single stranded tiles

Streptavidin-biotin marks



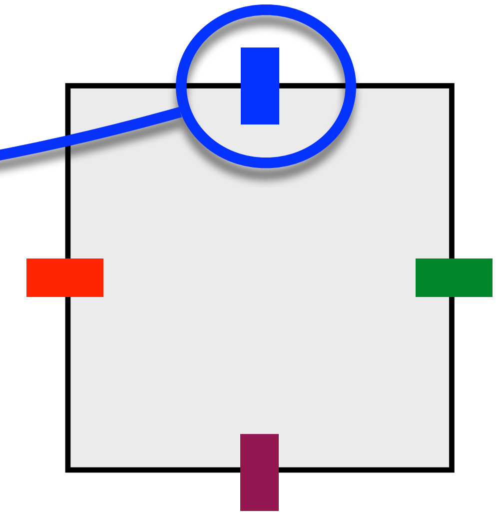
kTAM model for algorithmic assembly

Algorithmic self-assembly

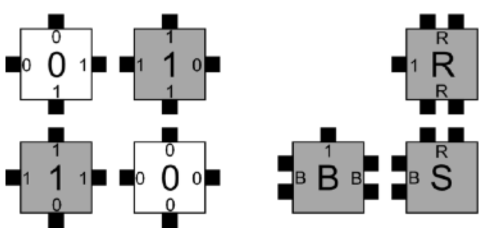


=

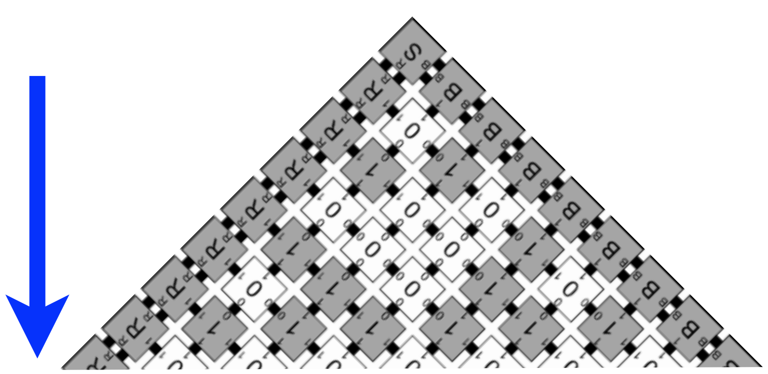
Winfree. 1998.
PhD Thesis



Erik Winfree had the idea that a growing lattice of DNA tiles could run a computer program, like Wang tiles or a CA

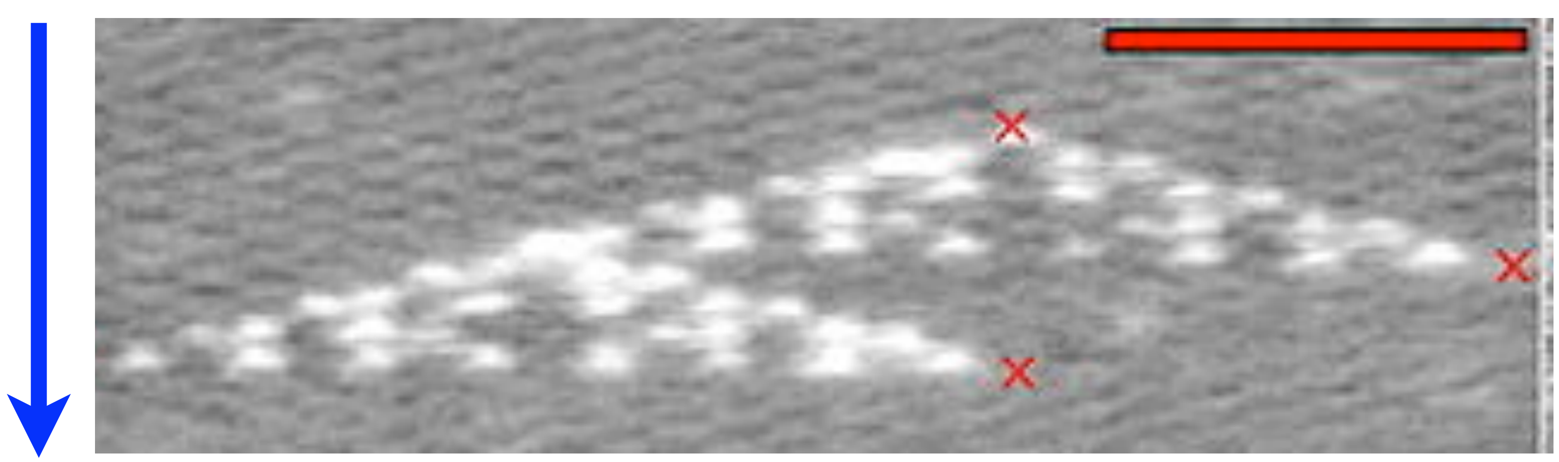


tiles = program



crystal growth =
program execution

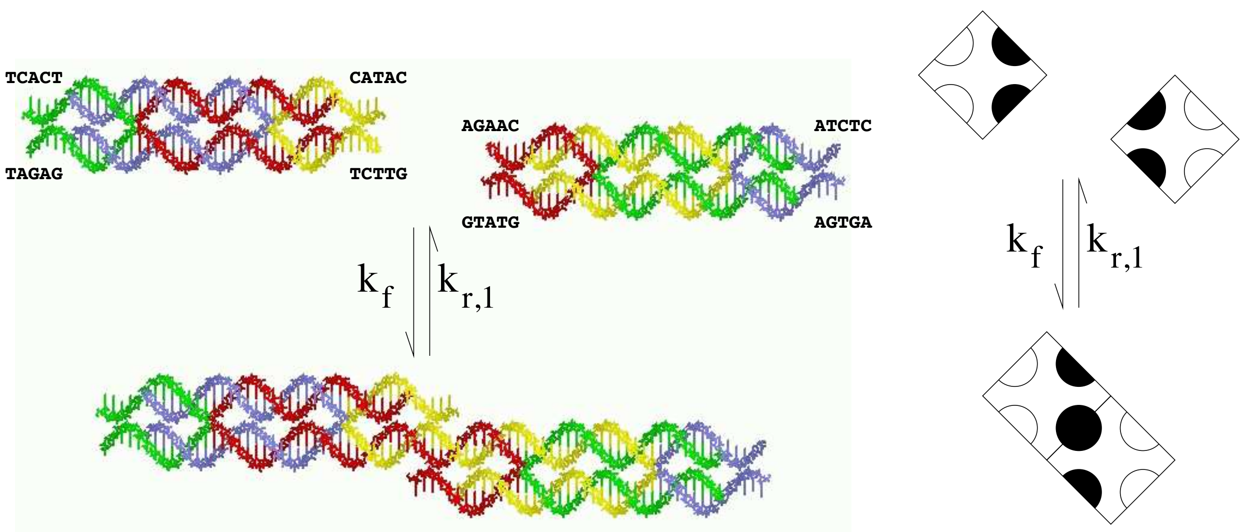
self-assembly.net



Rothemund, Papadakis, Winfree 2004

Thermodynamical model

(a)



Attachement rate

=

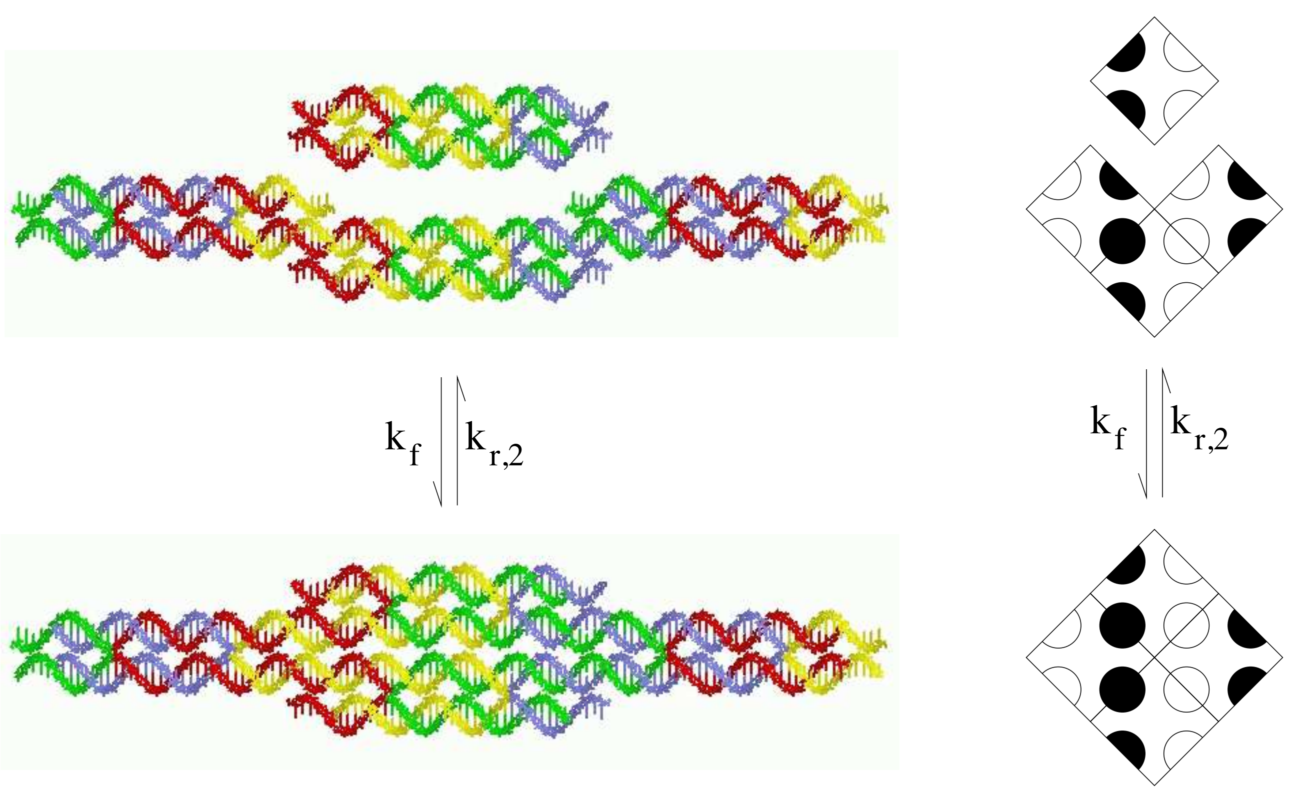
$$k_f \cdot [\, Strand \,]$$

$$=$$

$$k_f \cdot e^{-Gmc}$$

(mainly entropy)

(b)



Detachment rate

=

$$k_f \cdot e^{-(b \cdot Gse)}$$

where b is the number of bonds

and $G_{se} = \Delta G / RT$

the bonding unit energy in RT units

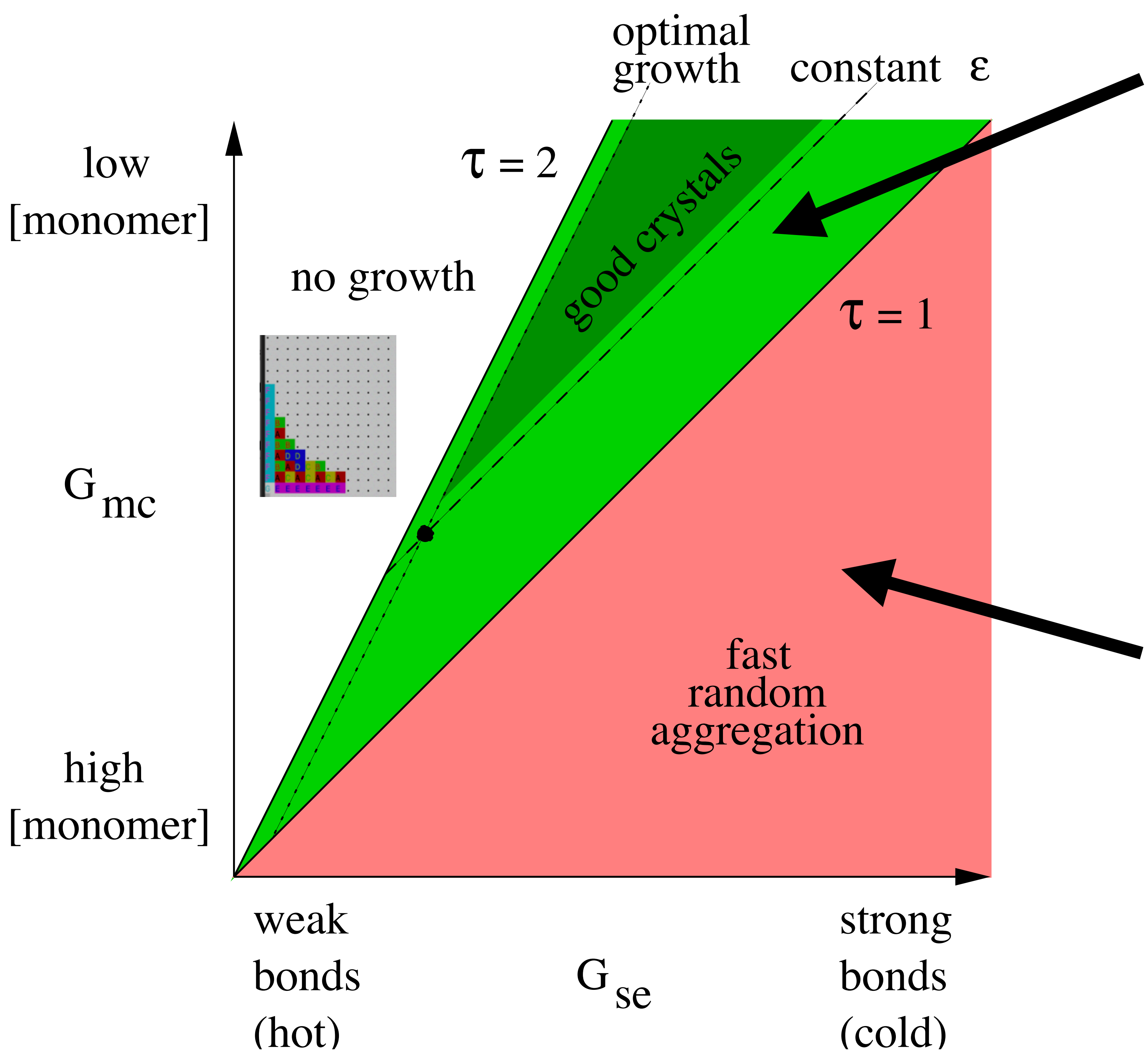
(mix of entropy and enthalpy)

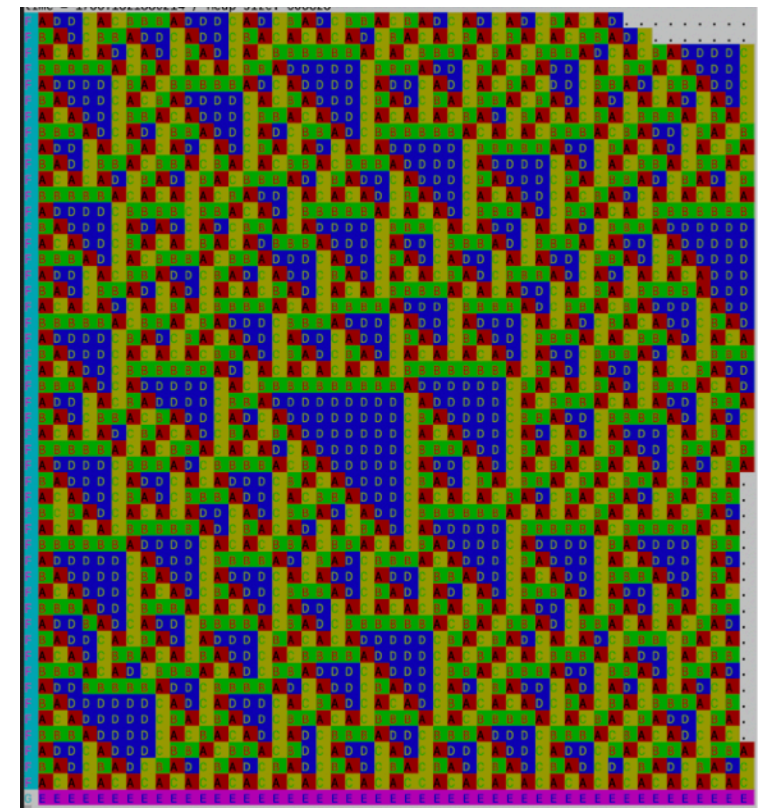
mc = **m**onomer **c**oncentration

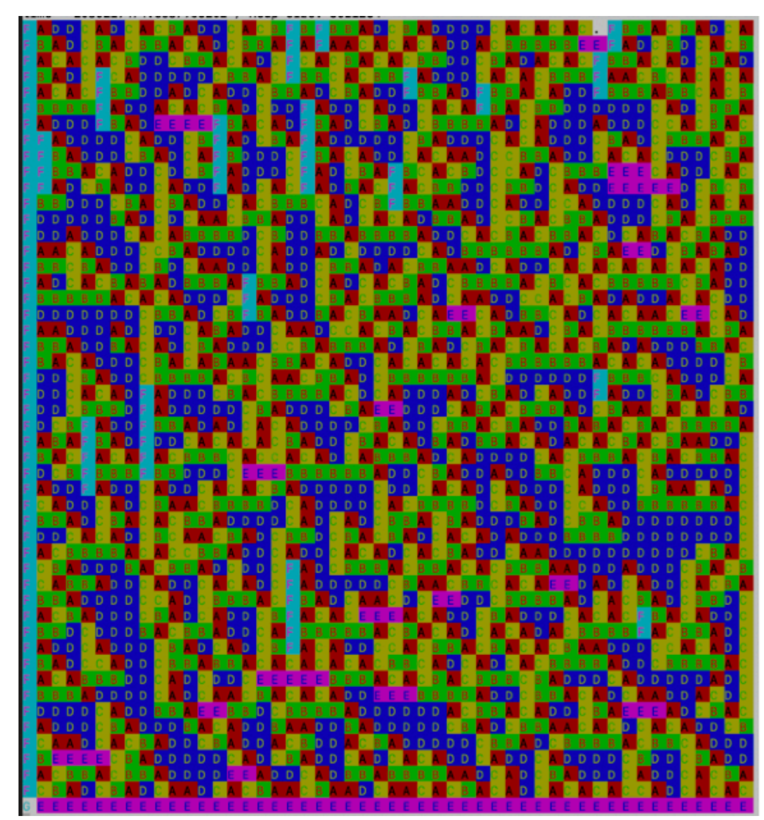
se = **s**ticky **e**nd bond strength

Winfree, Bekbolatov DNA9

Simulations

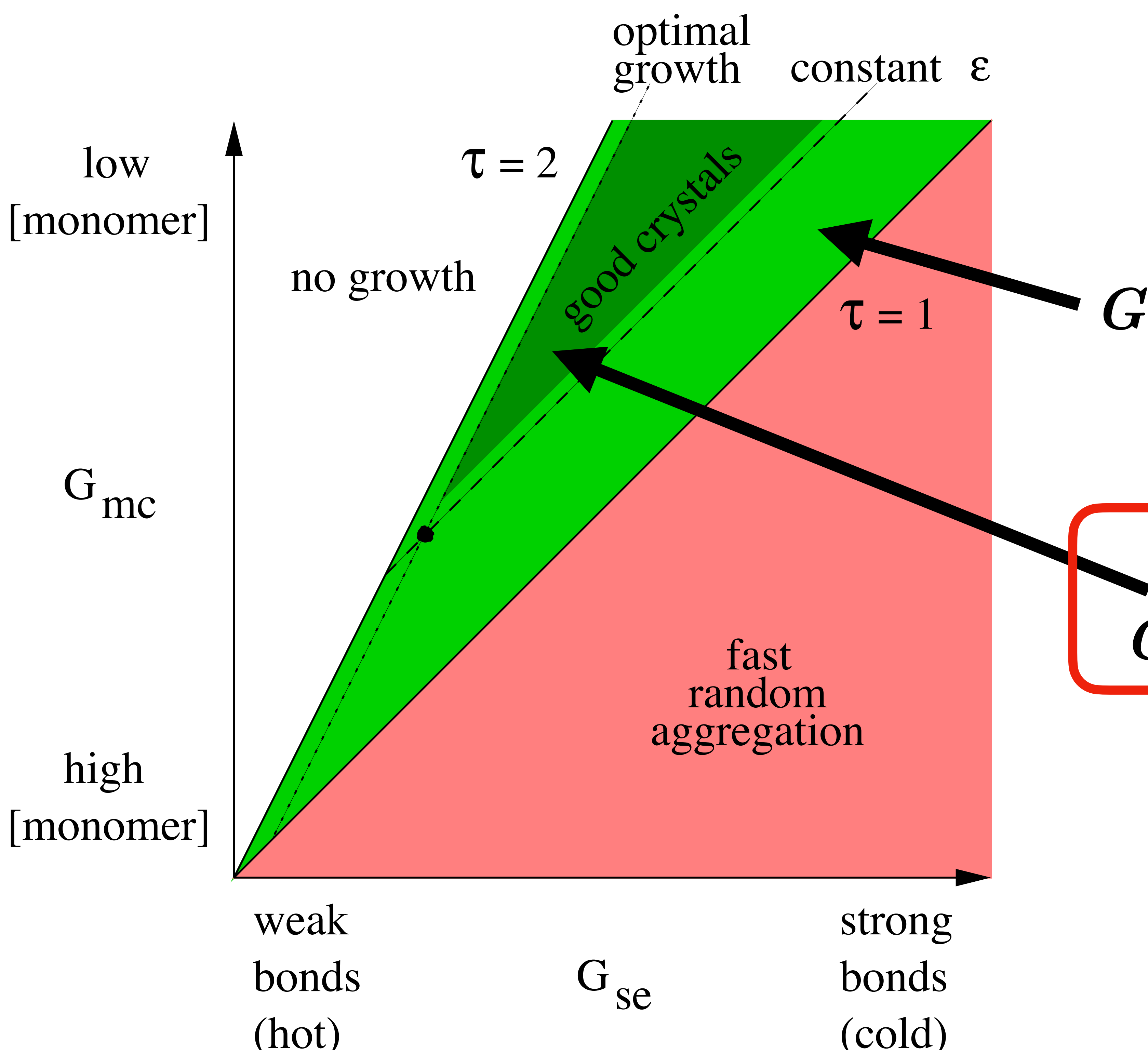






Winfree, Bekbolatov DNA9

Simulations



$G_{se} < G_{mc} < 2G_{se}$

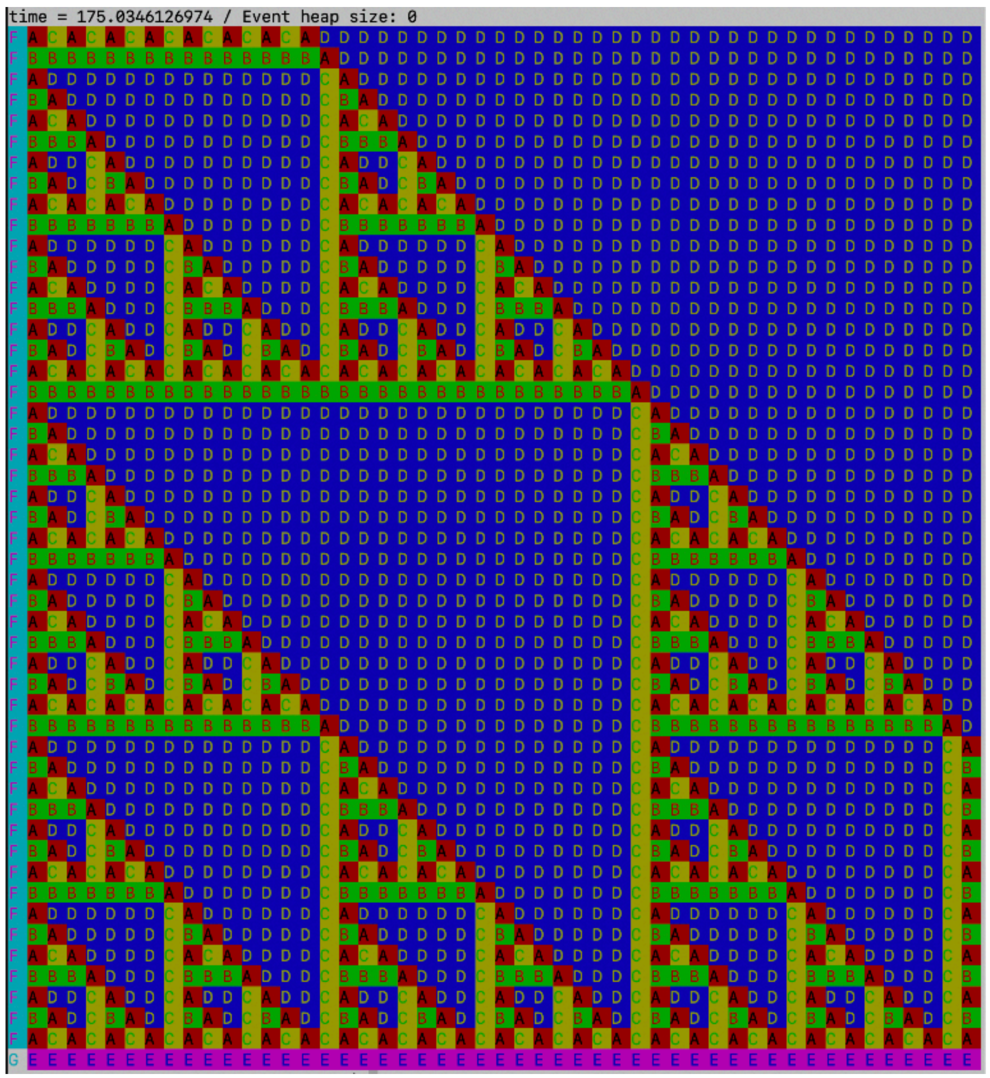
Fastest when

$G_{mc} \sim 2G_{se} - \varepsilon$

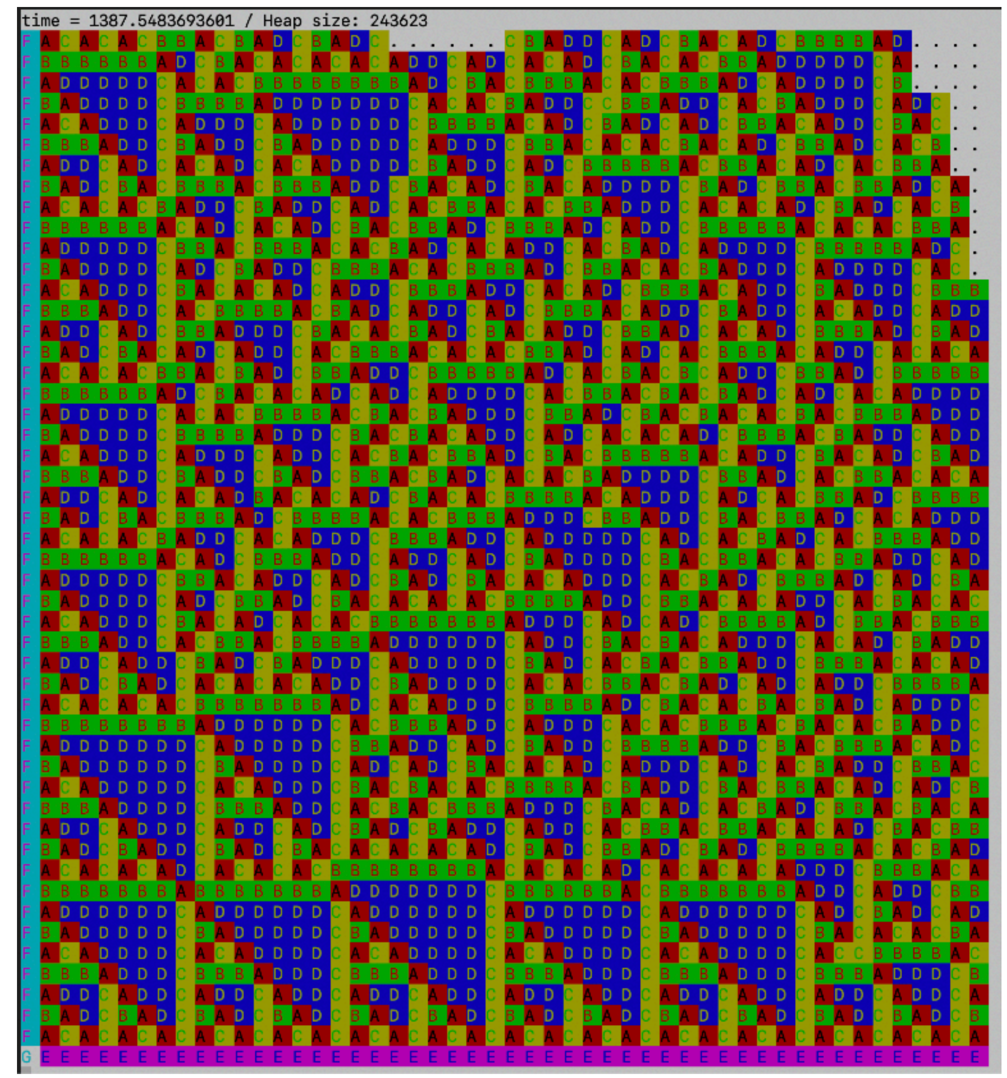
Sweet spot

Winfree, Bekbolatov DNA9

Minimizing errors

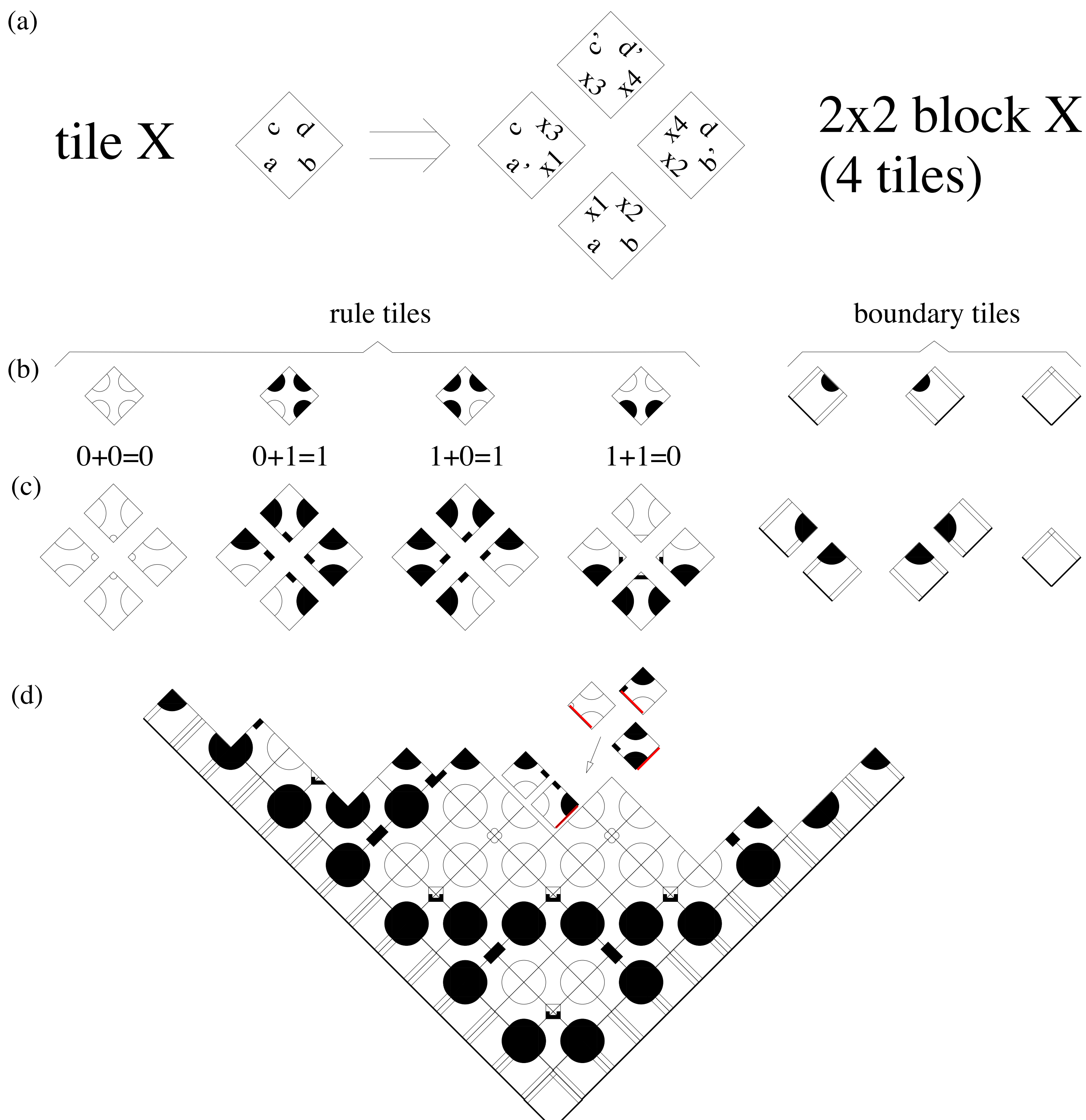


Desired



Obtained

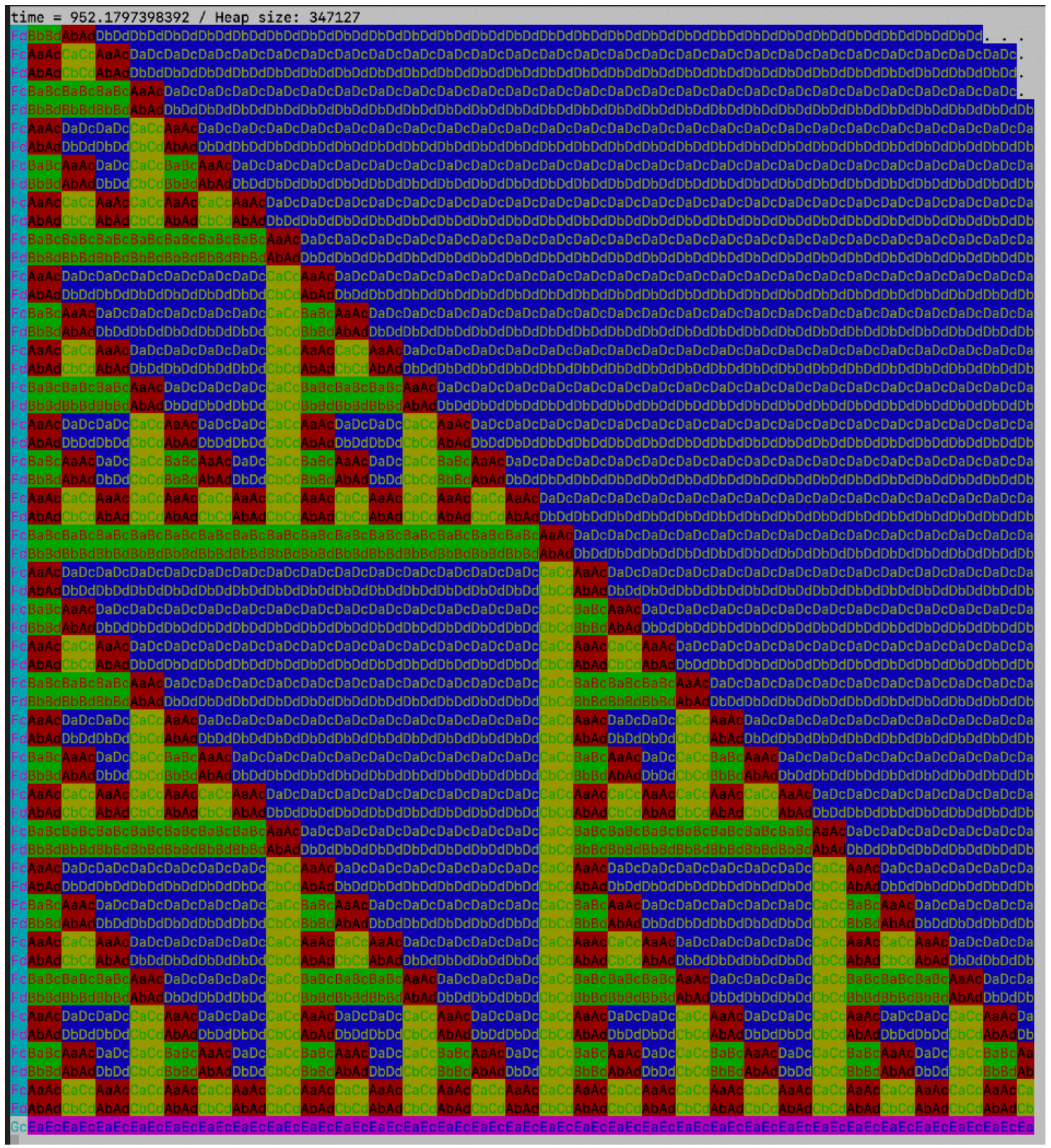
Proofreading tiles



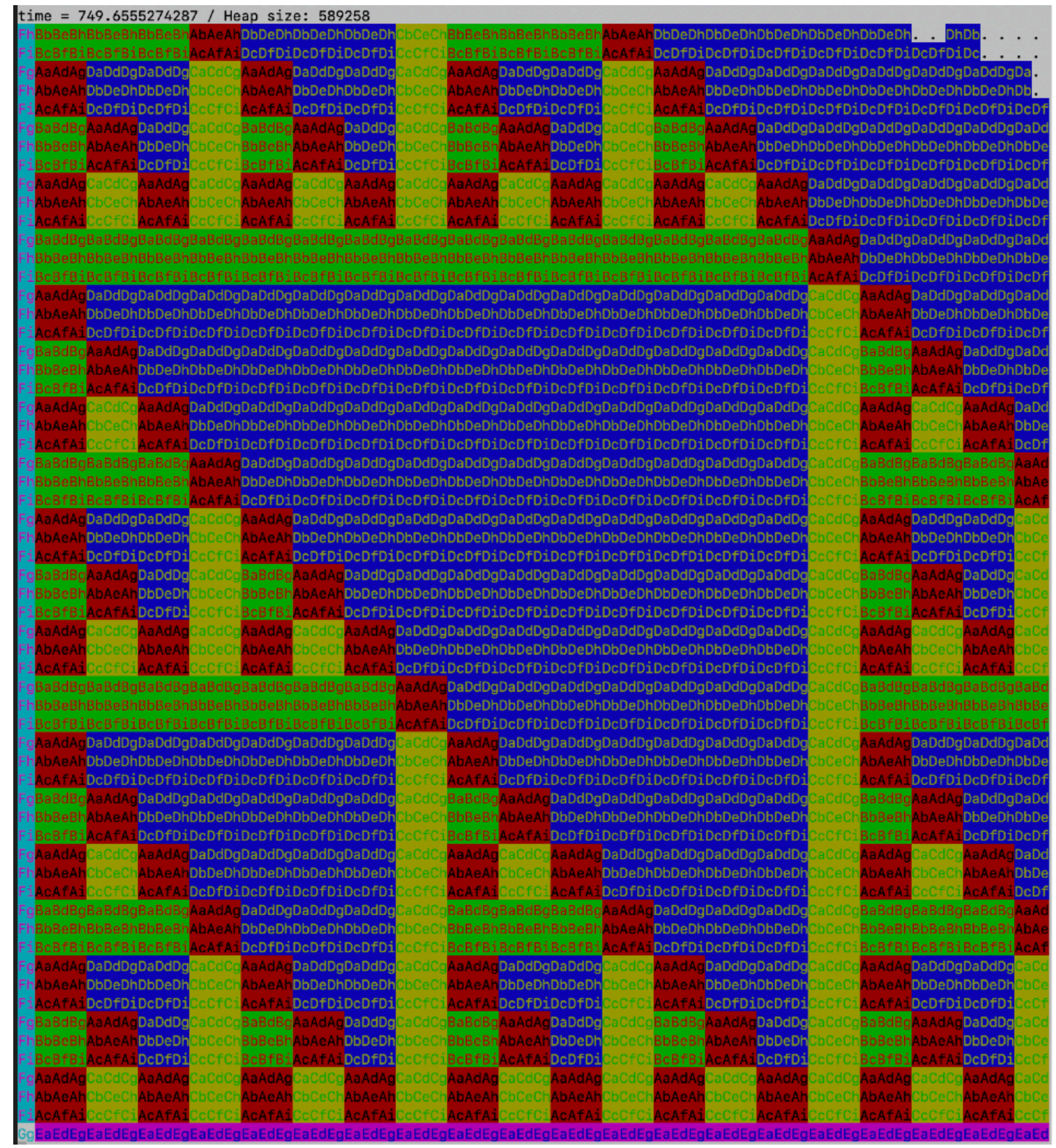
- Cut every tile into k x k tiles
- Now, you need to ***make an other error*** to compensate for an error
- The error rate is squared for $k = 2$!

Winfree, Bekbolatov DNA9

Proofreading tiles

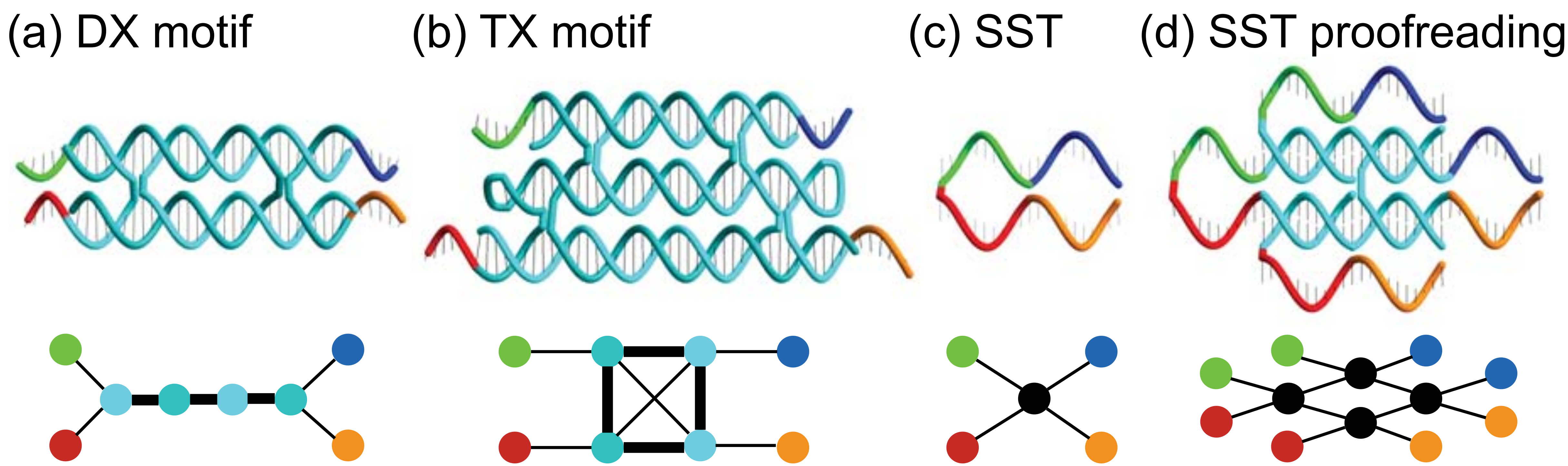


$k = 2$



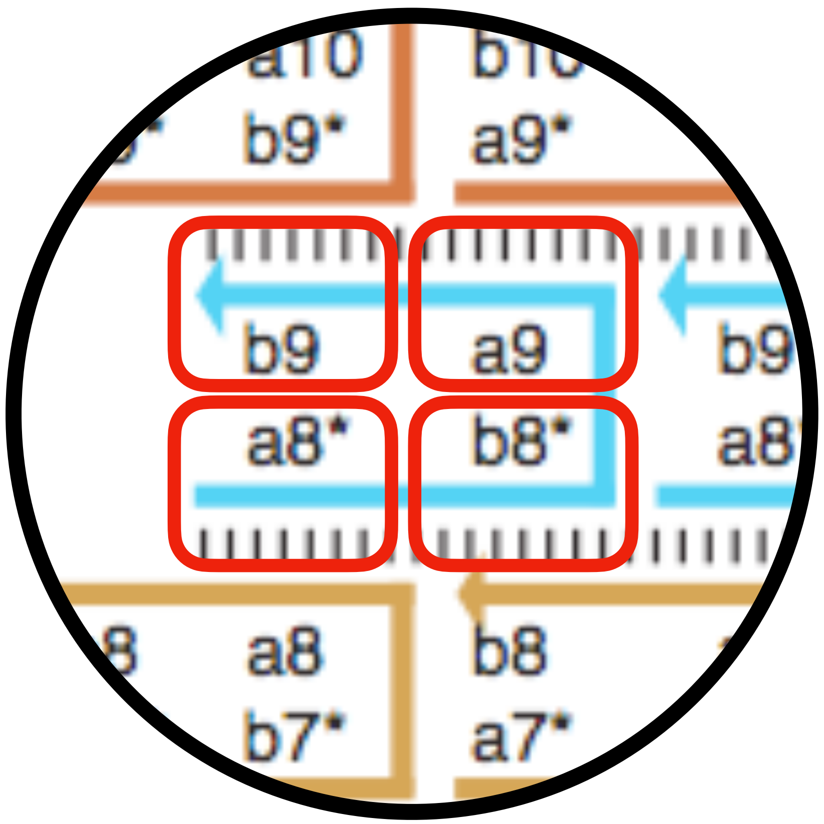
$k = 3$

Proofreading tiles compared to other tiles



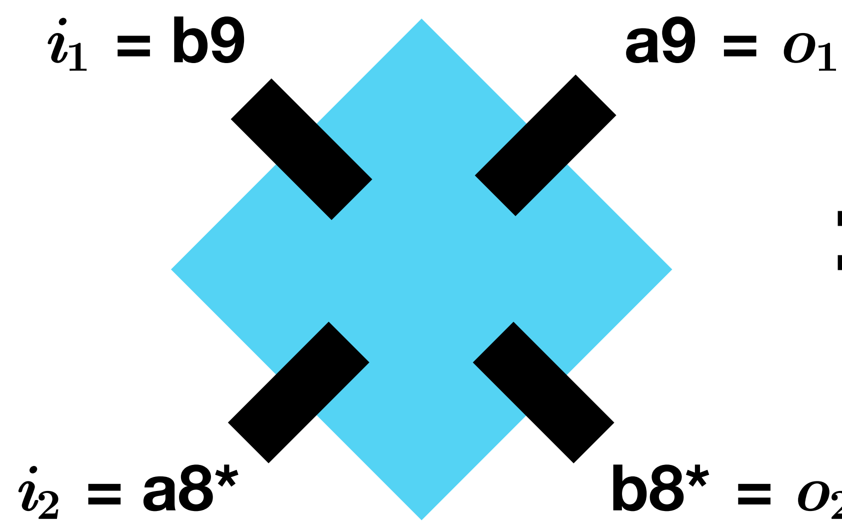
Implementing boolean circuits

Tile as gates

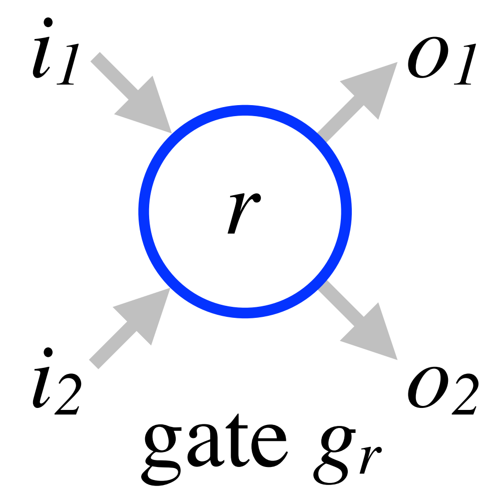


4 domains = 4 glues

=

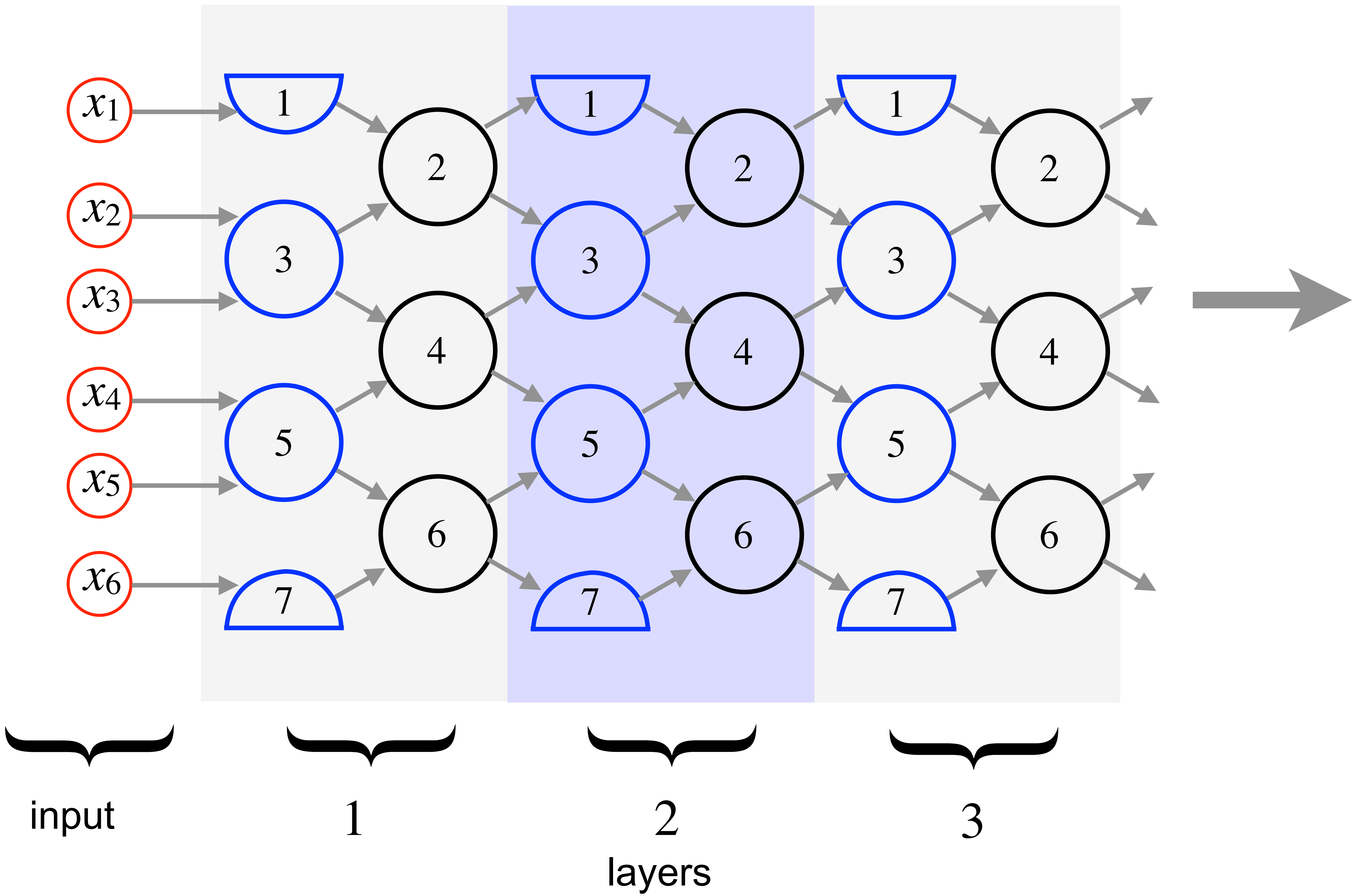


=

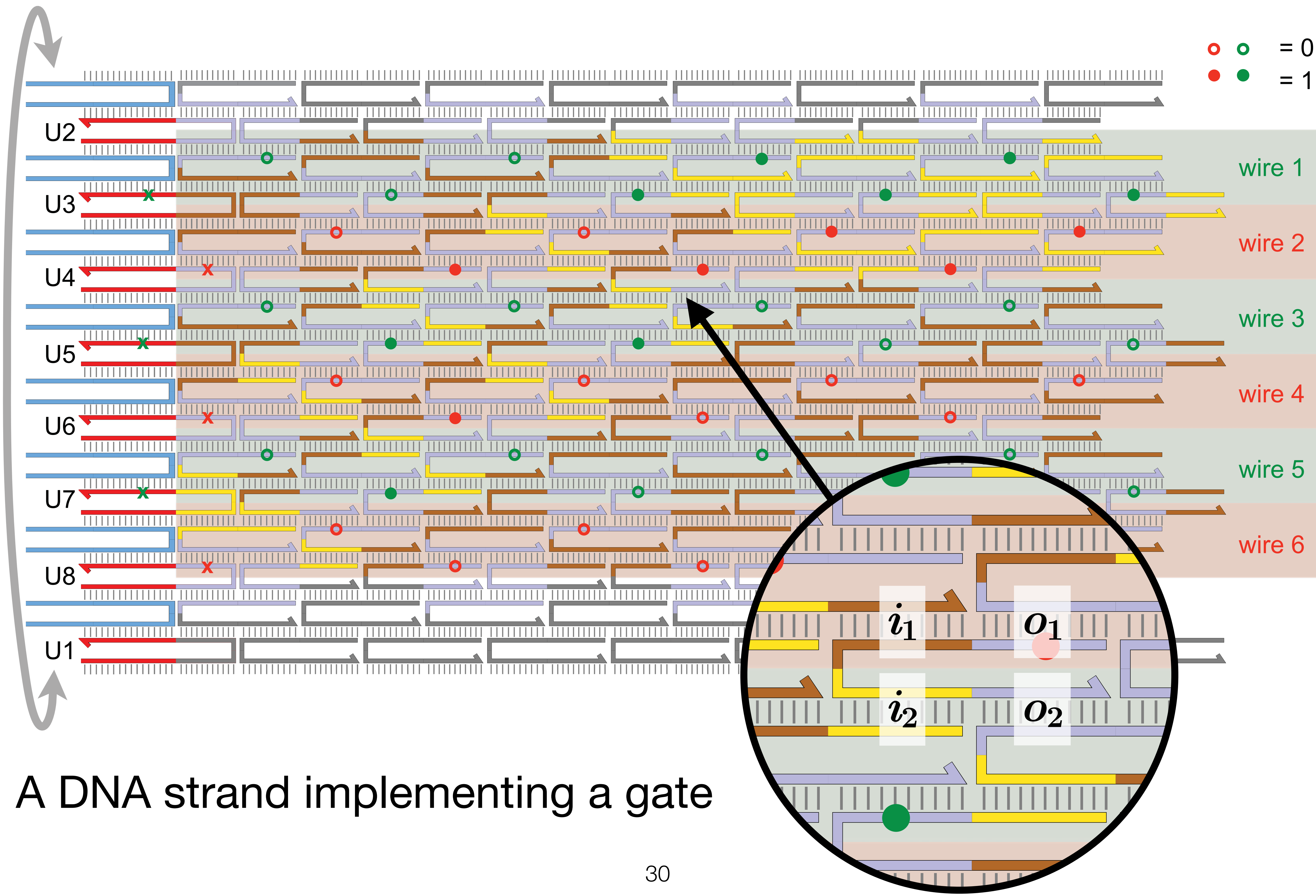


Tiles assembly is a rewriting system

DNA nanotube circuit model

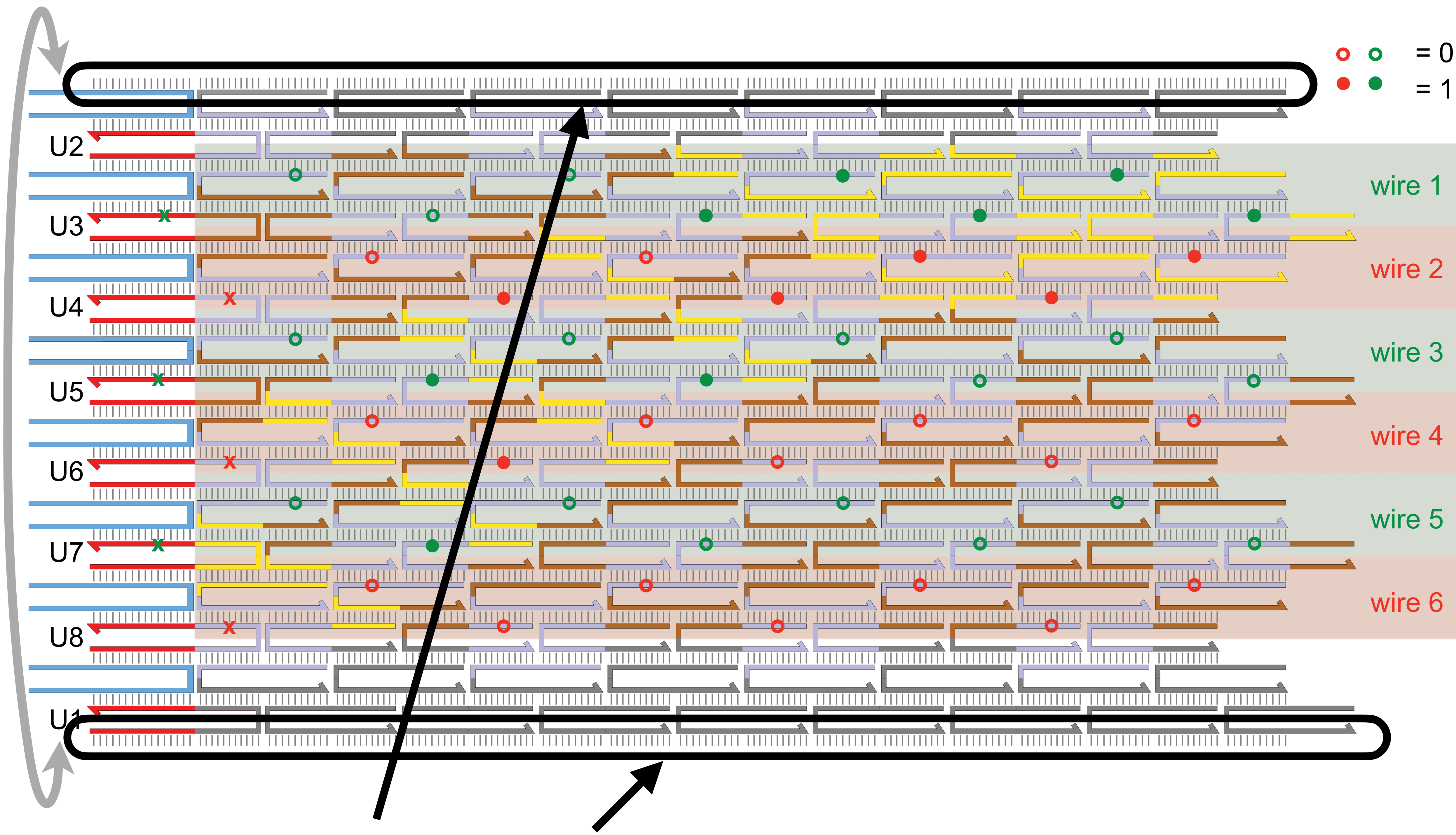


DNA nanotube circuit model



A DNA strand implementing a gate

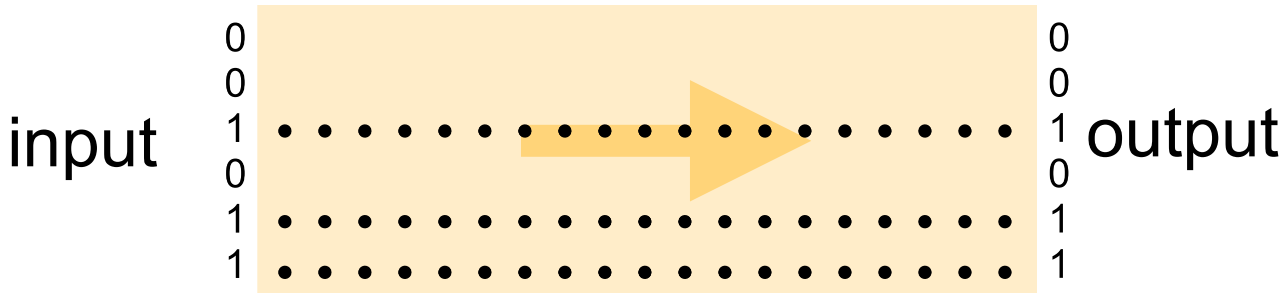
DNA nanotube circuit model



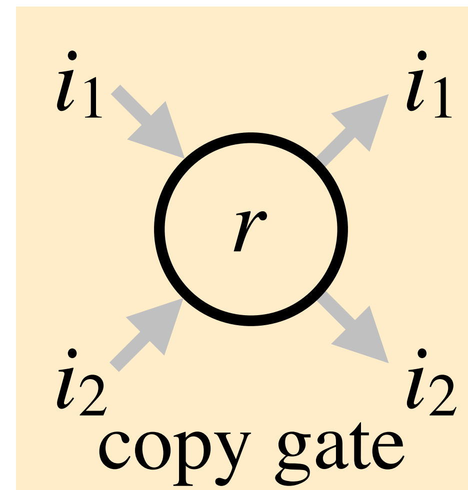
The *seam* which can be *unzipped* to flatten the assembly for imaging

Example nanotube circuits

- ***n*-bit copying**: n+1 copy gates



6-bit copying circuit

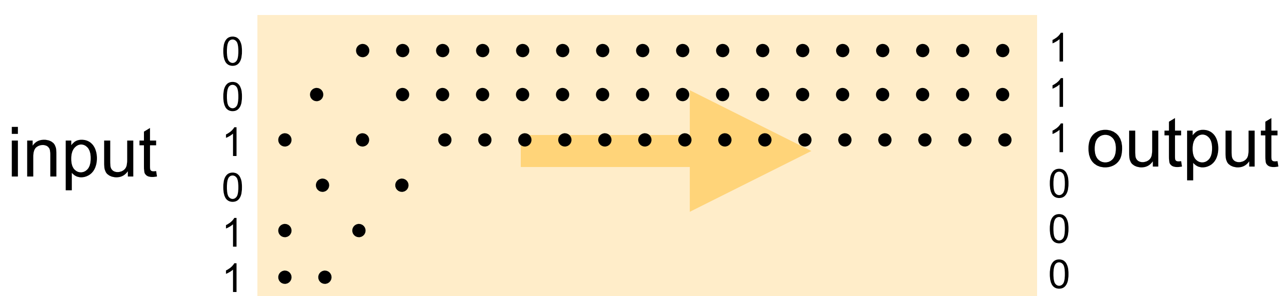


$i_1, i_2 \in \{0,1\}$

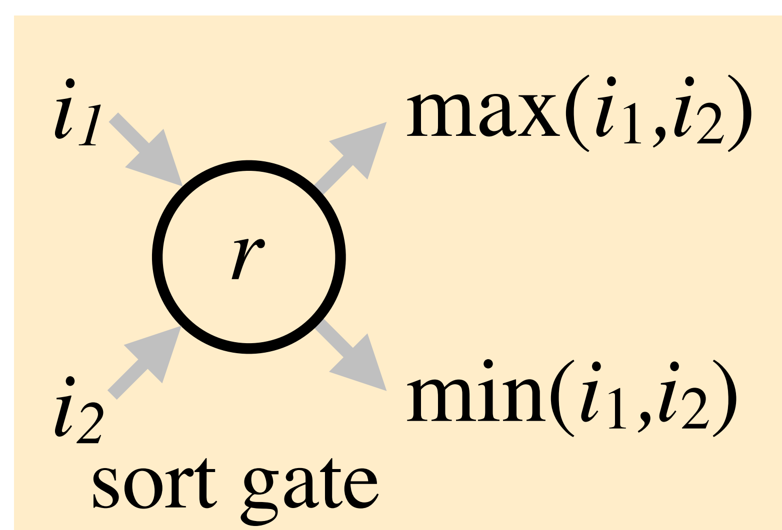
i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

copy gate truth table

- ***n*-bit binary sorting:** n+1 sort gates



6-bit sorting circuit



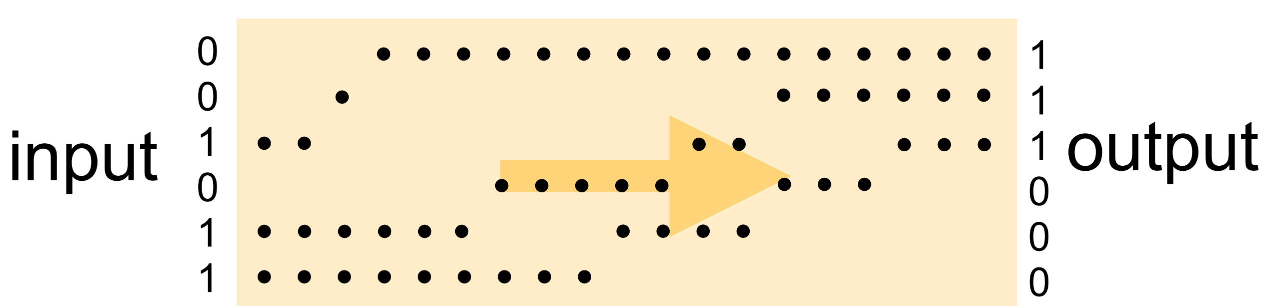
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

sort gate truth table

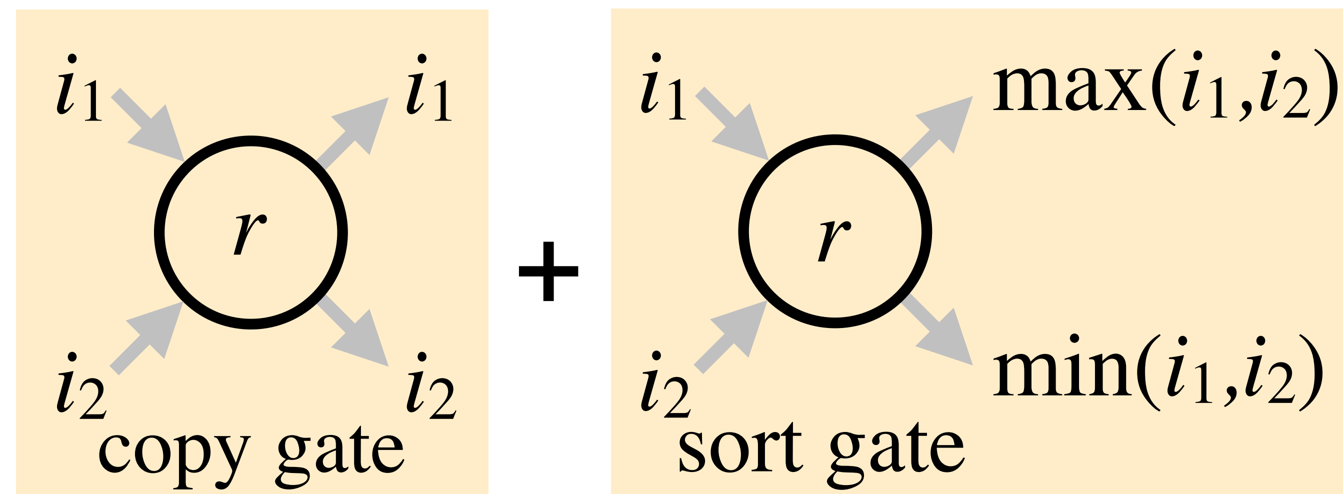
Note that 2 gates are single input, single output

Example nanotube circuits

- **Lazy sorting**! Take the union of the copy gate set and the sort gate set. Copying fights to slow down the sorting process, but assuming a fair execution, sorting will eventually win.



6-bit slow randomised sorting

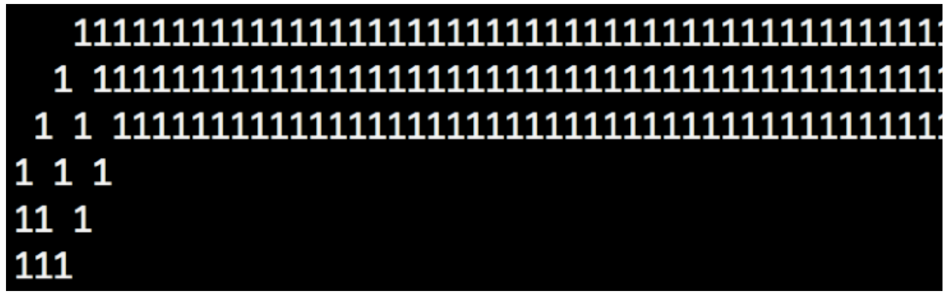


- Since, in any given circuit, each gate "knows" its row number *r*, we will also write circuits (programs) that exploit this feature, do something that is interesting *and* (more importantly) provably impossible without that feature

Circuits

sorting

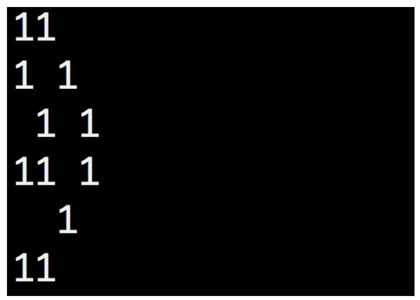
0
0
0
1
1
1



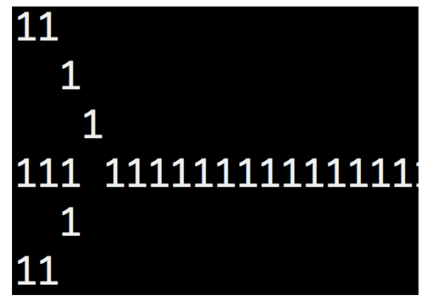
Function computation

parity

1
1
0
1
0
1



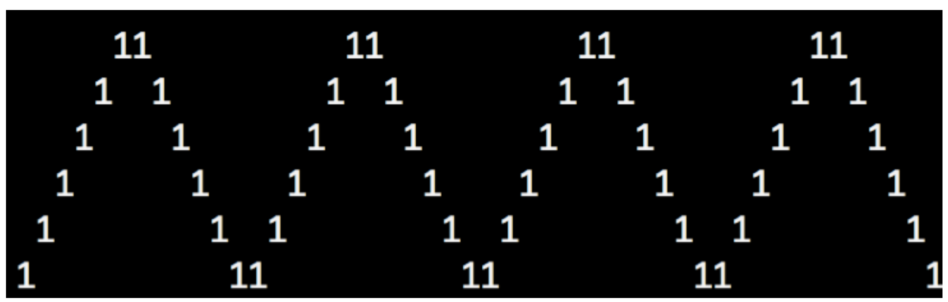
1
0
0
1
0
1



Solving a “hard” decision problem

zig-zag

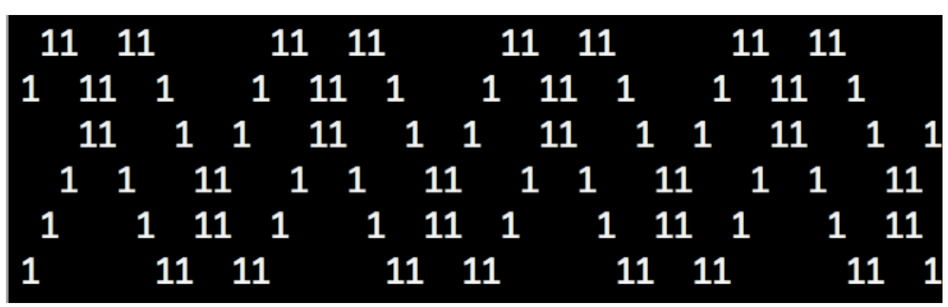
0
0
0
0
0
1



Glider: A common cellular automata primitive

zig-zag

0
1
0
0
0
1



Pattern: DNA

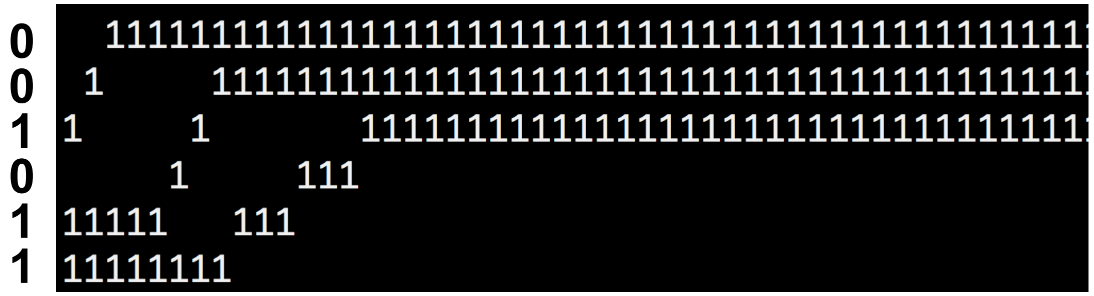
long repeat

Behaviour: 63 layers to see the same thing twice!

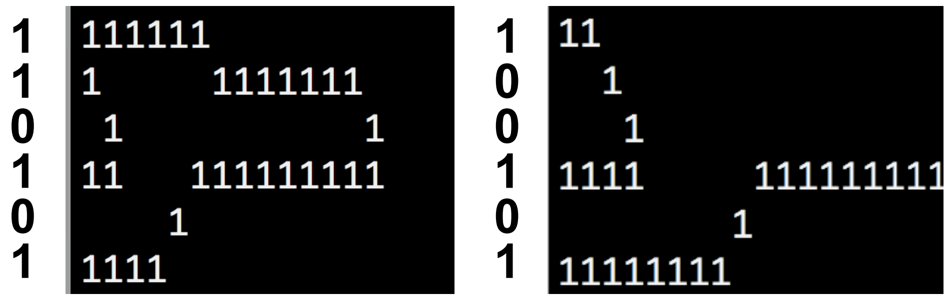
Rule 110

Circuits: randomised

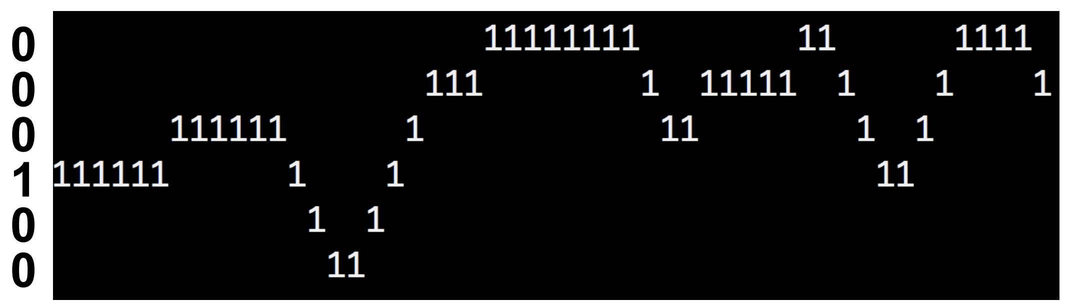
lazy sorting



lazy
parity
to-the-middle

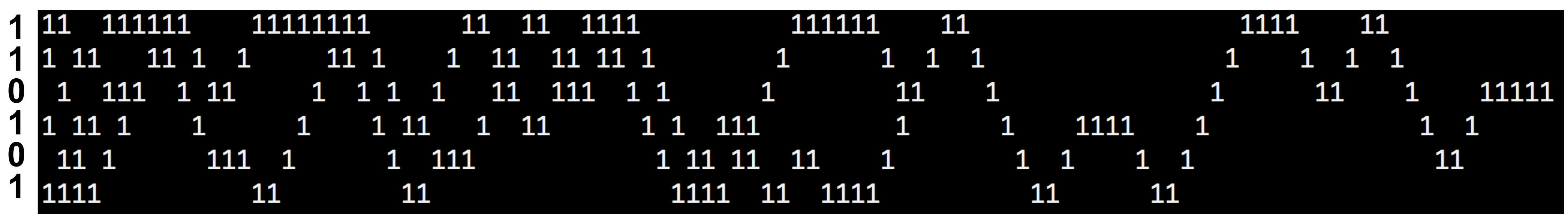


random
walking bit



Also, rand walk,
absorbing at edge

leader
election
under randomised
bit walking



problem studied in cellular automata, distributed computing, networks, CRNs

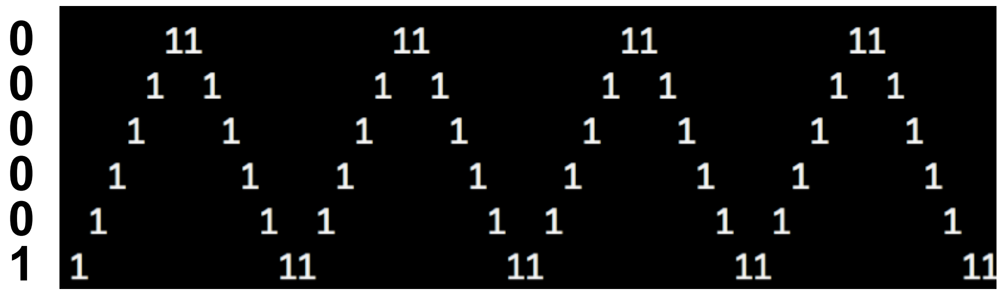
fair coin

Randomised programs may be a useful tool to calculate energetics of tile binding, or groups of tiles binding, from AFM data

A nice method to assess the quality of our sequence design

Circuits

zig-zag

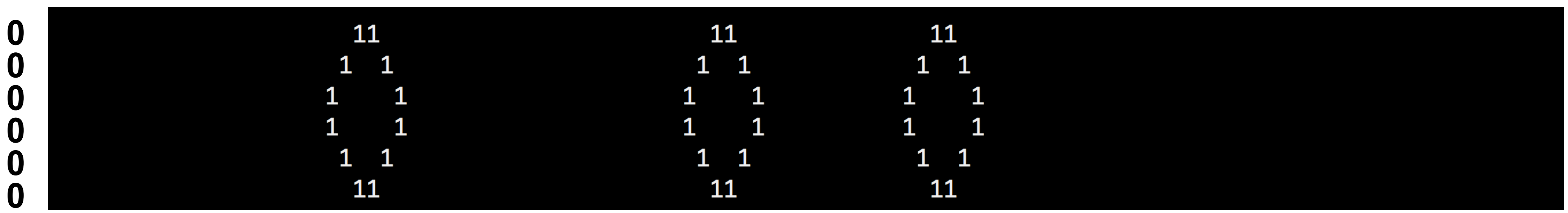


Glider: A common cellular automata primitive

Pattern: Monotone / horizontally connected

Nonmontonic widely-spaced patterns are provably impossible in the deterministic circuit model

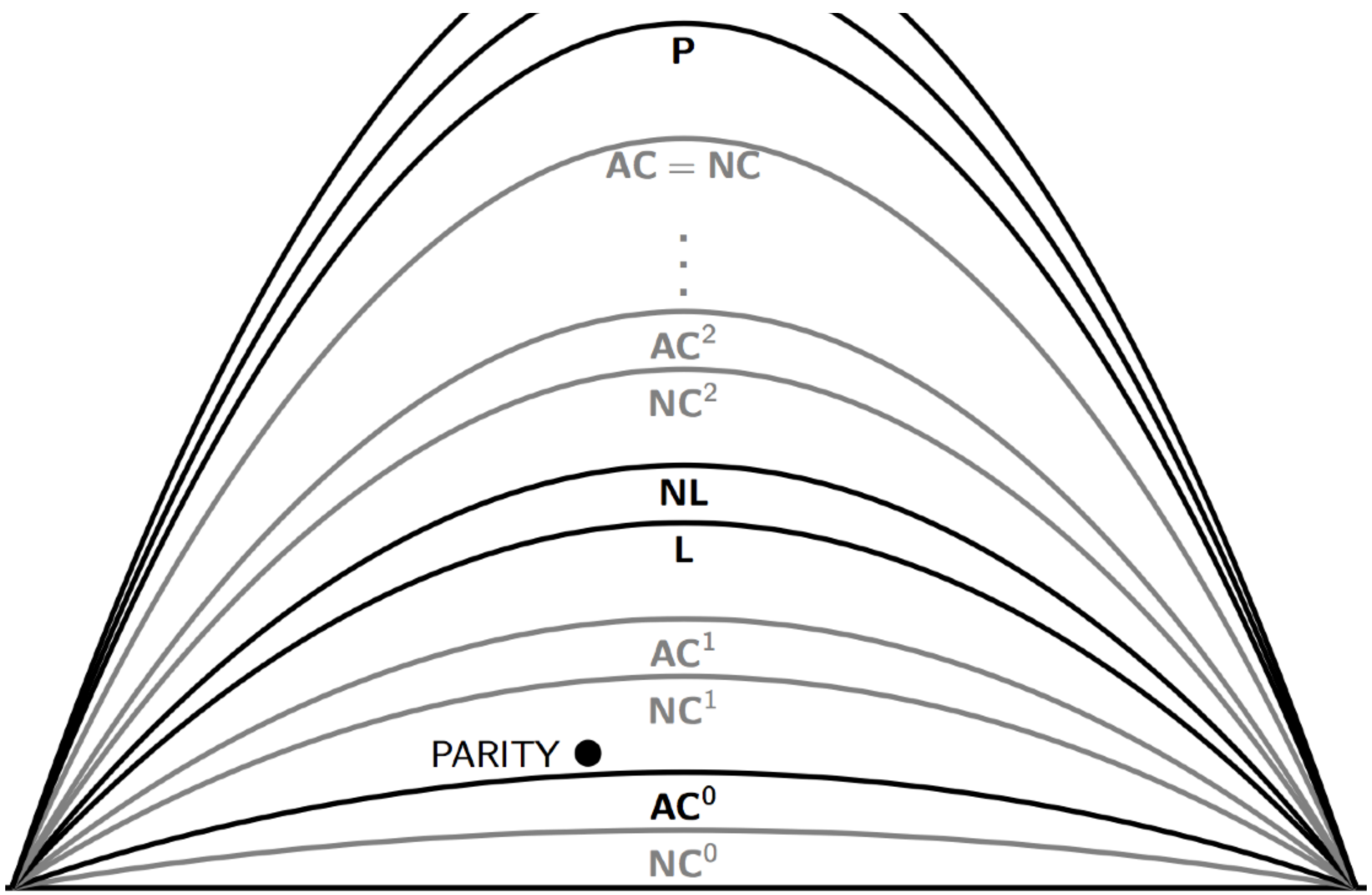
Diamonds are forever



Blowing bubbles

Computational power of DNA (DNA = DNA nanotube algorithms)

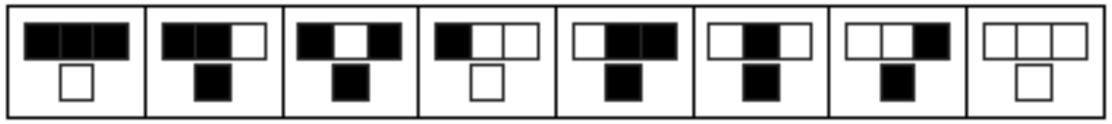
- What is the computational power of our circuit model?
- With *n* input bits, depth-2 layer, and poly(n) depth circuit, what can be solved?
  - No more than P (proof: simulate poly(n) depth circuit in polynomial time on a Turing machine)
  - We've seen already that the model can solve SORTING, PARITY both of which are outside AC^0

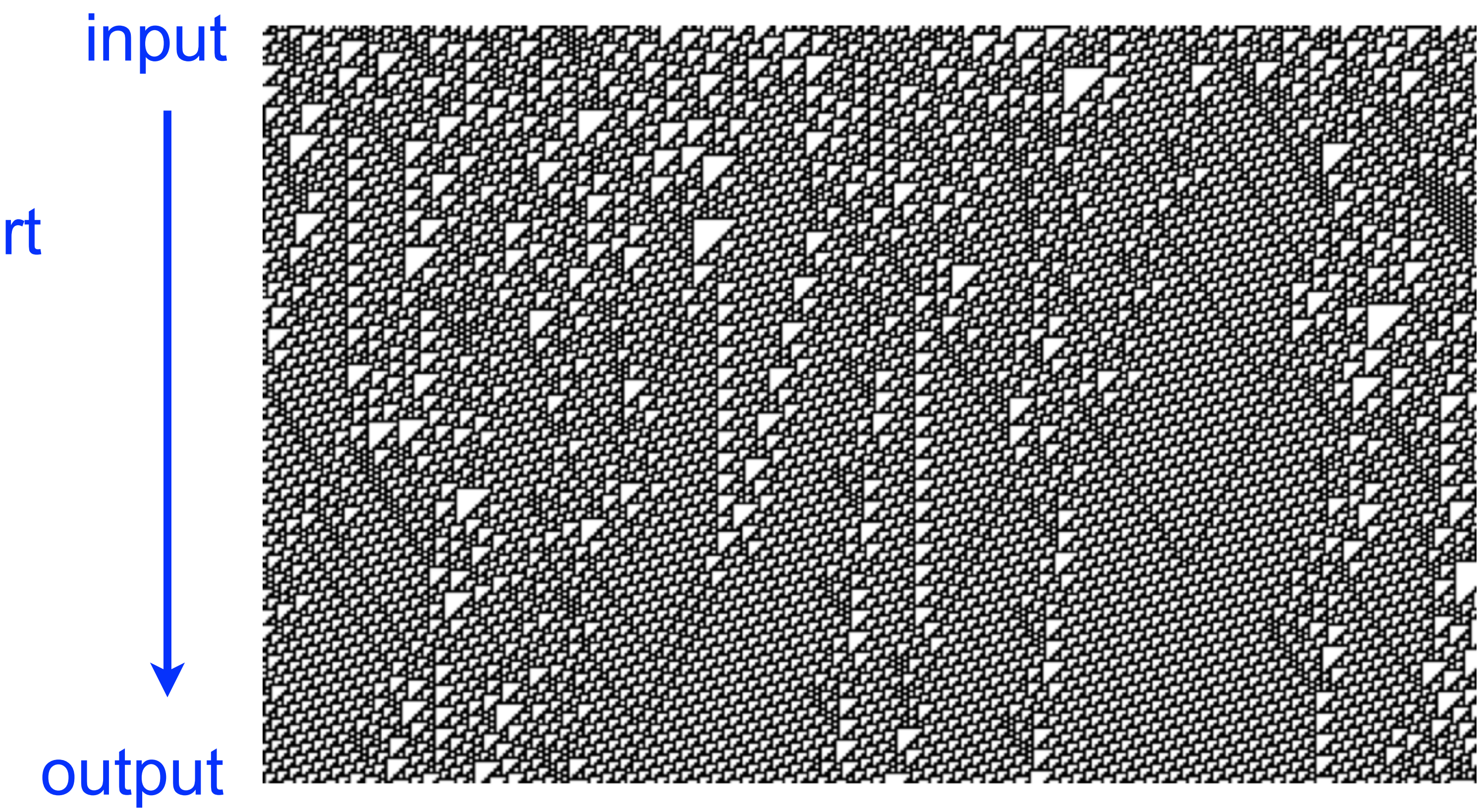


?

Rule 110

Rule 110





- **Theorem:** Rule 110 is an **efficient** and **general purpose** computer

Neary, Woods. ICALP 2006

Cook. Complex Systems. 15:1-40 2004

Computational power of DNA
(DNA = DNA nanotube algorithms)

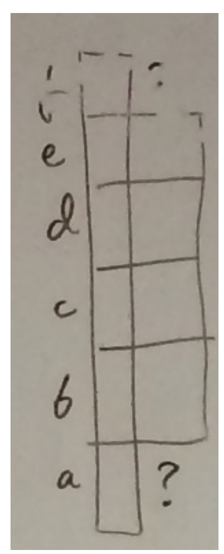
- What is the computational power of our circuit model?
- With *n* input bits, depth-2 layer, and poly(n) depth circuit, what can be solved?
  - No more than P. Proof: simulate poly(n) depth circuit in polynomial time on a Turing machine
  - All of P: Proof: simulate Rule 110

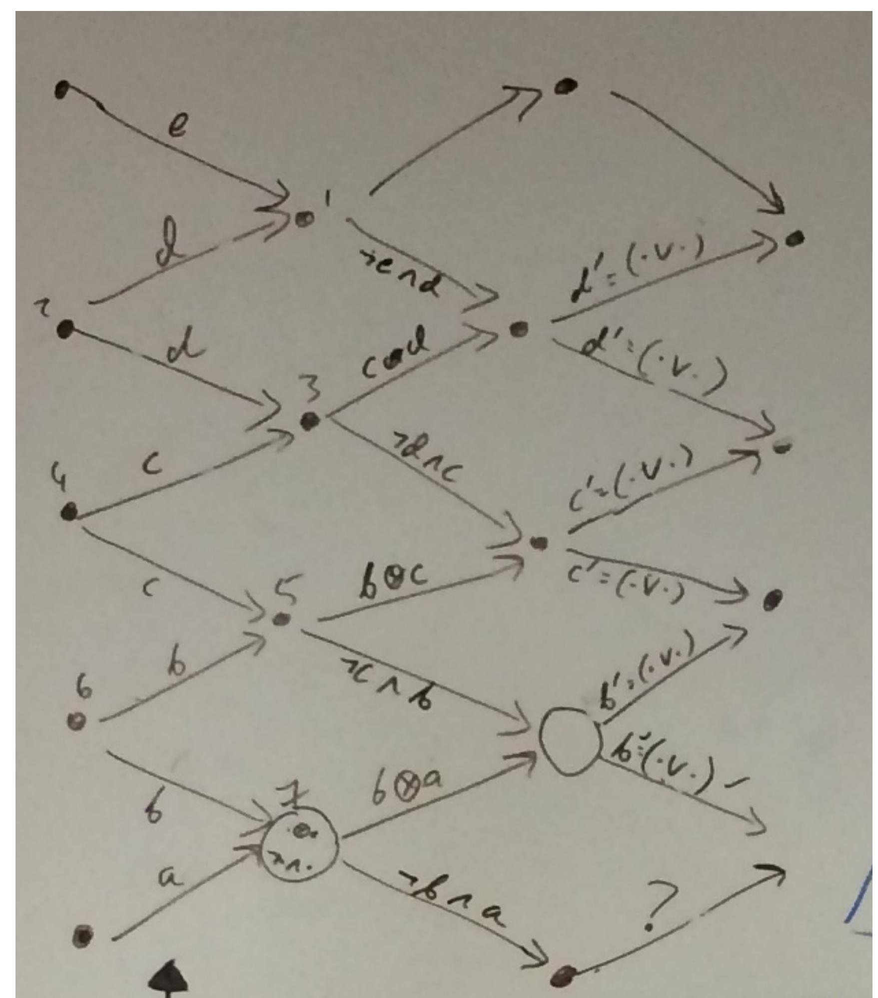
c b a

$$F(0,0,0) = 0$$
$$F(0,0,1) = 1$$
$$F(0,1,0) = 1$$
$$F(0,1,1) = 1$$

c b a

$$F(1,0,0) = 0$$
$$F(1,0,1) = 1$$
$$F(1,1,0) = 1$$
$$F(1,1,1) = 0$$

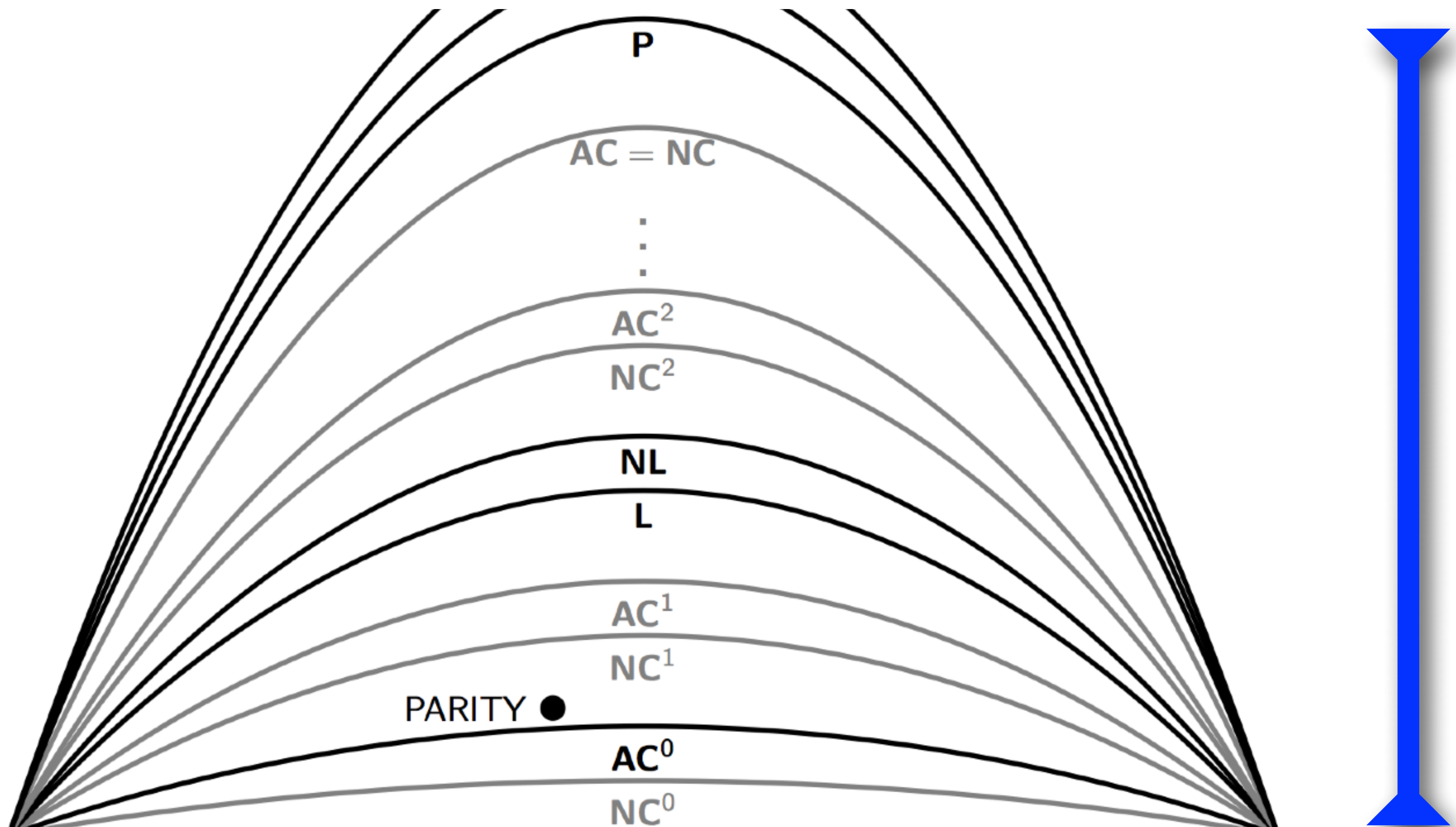




Computational power of DNA (DNA = DNA nanotube algorithms)

- What is the computational power of our circuit model?
- With *n* input bits, depth-2 layer, and poly(n) depth circuit, what can be solved?
  - Answer: Exactly P, via Rule 110 simulation

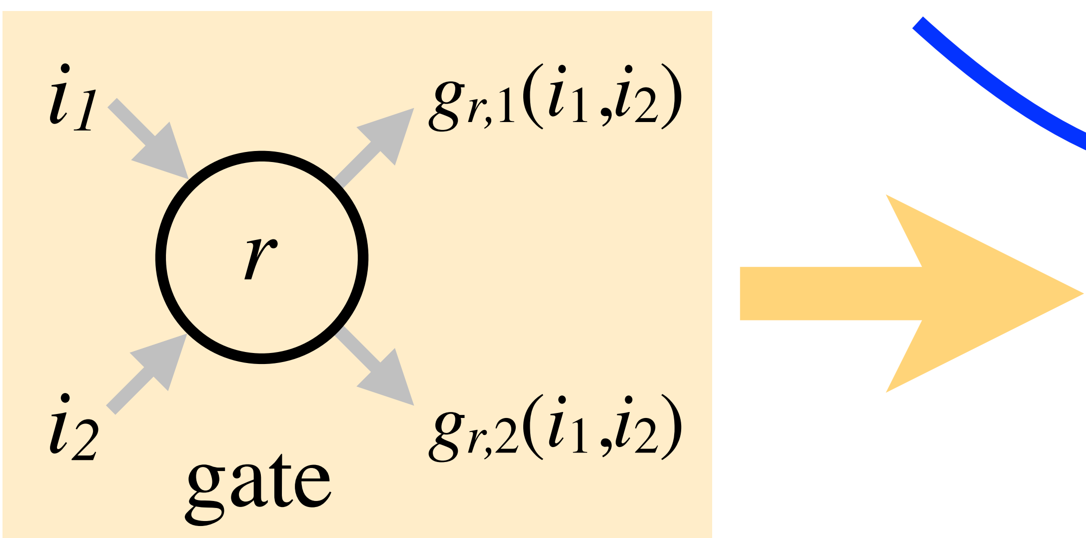
T. Neary, D. Woods. P-completeness of cellular automaton Rule 110. ICALP 2006. Springer LNCS 4051(1):132-143
Cook, M.: Universality in elementary cellular automata. Complex Systems 15 (2004) 1–40



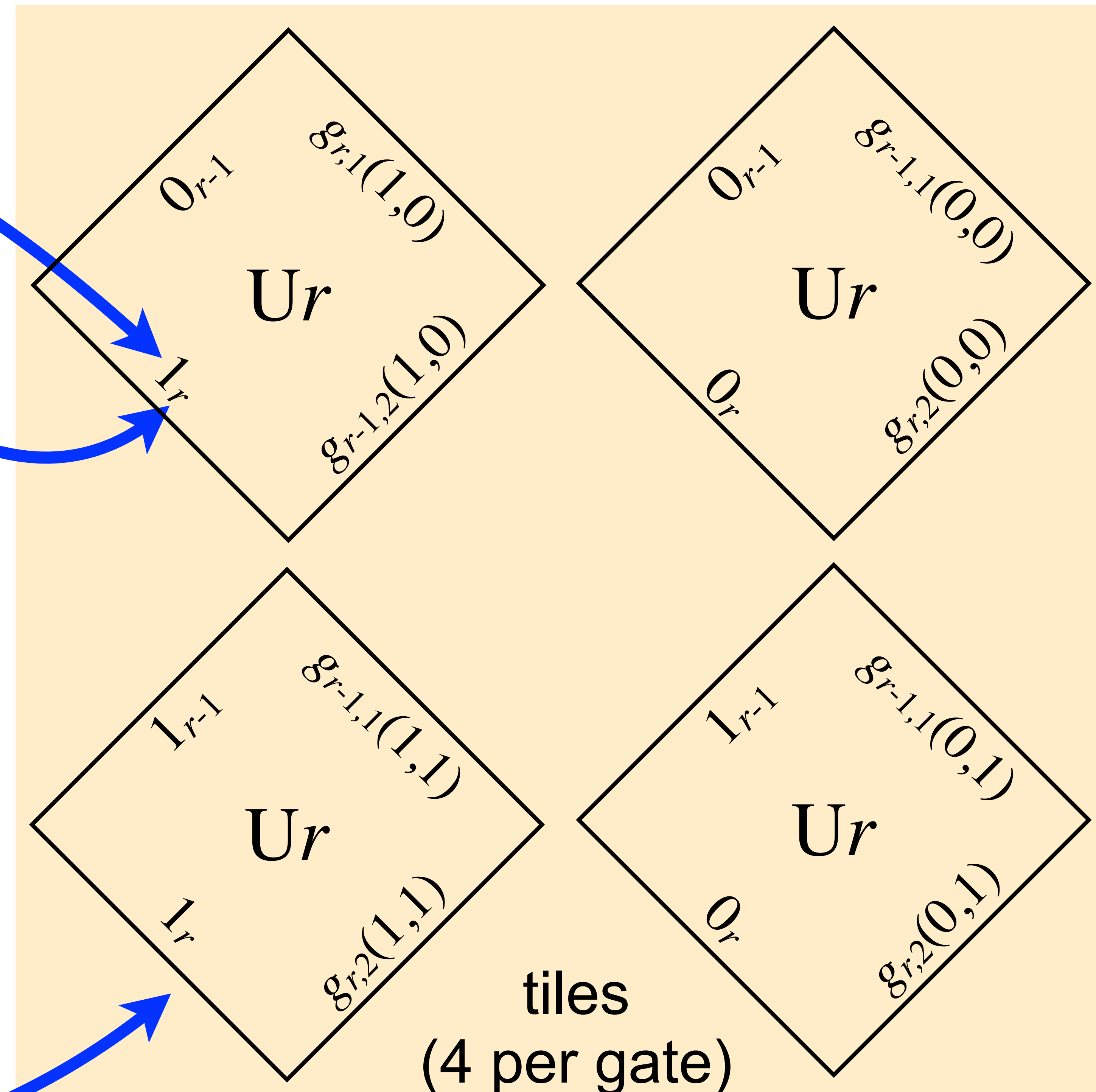
From gate abstraction to tile abstraction

1. Compile gates to tiles

Each **glue** encodes a **bit** and a **layer row**



$i_1, i_2, g_{r,1}(i_1,i_2), g_{r,2}(i_1,i_2) \in \{0,1\}$

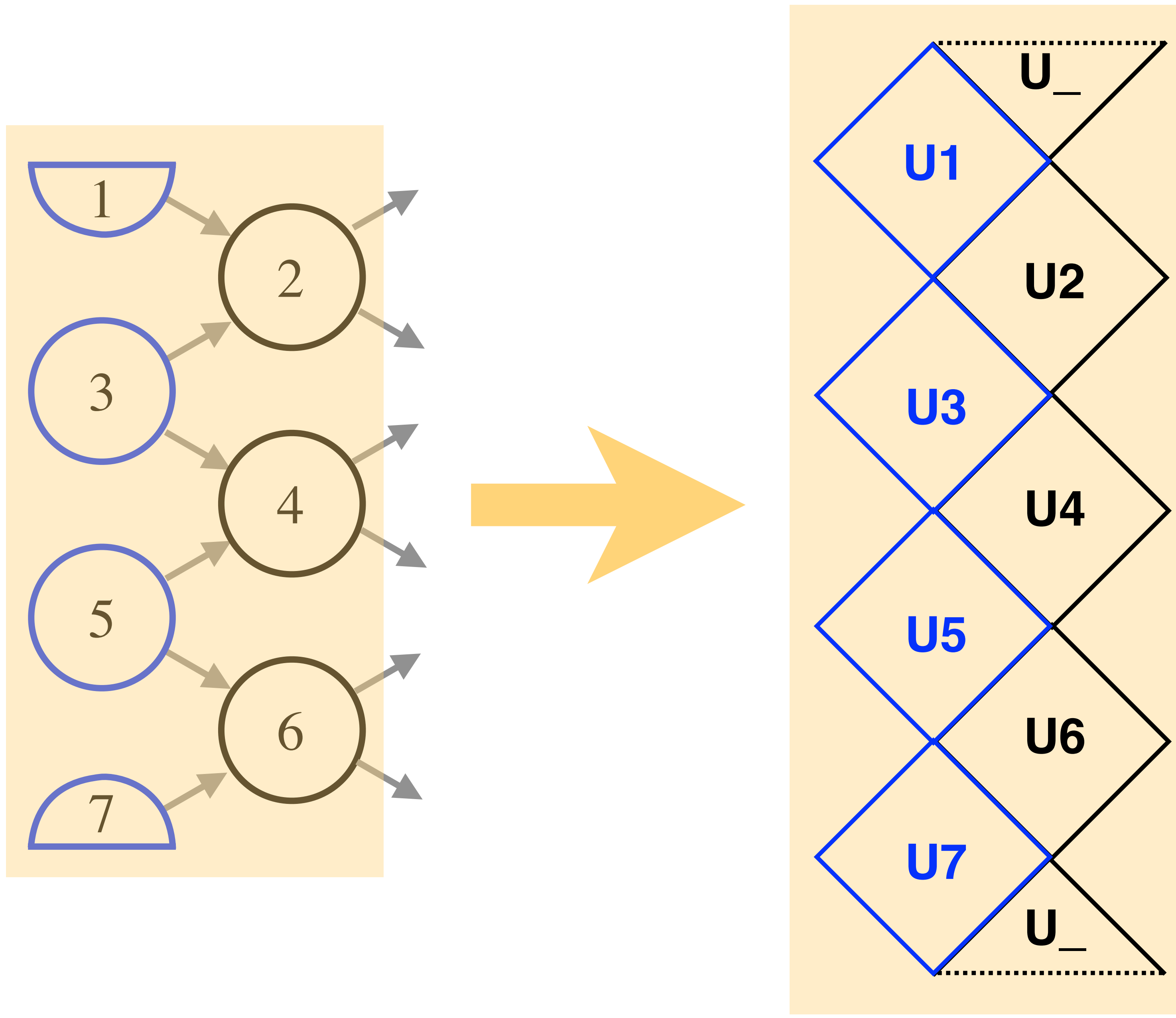


i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Gate truth table

Each **row** of a gate's truth table is encoded by a **tile**

6-bit universal tileset: overview

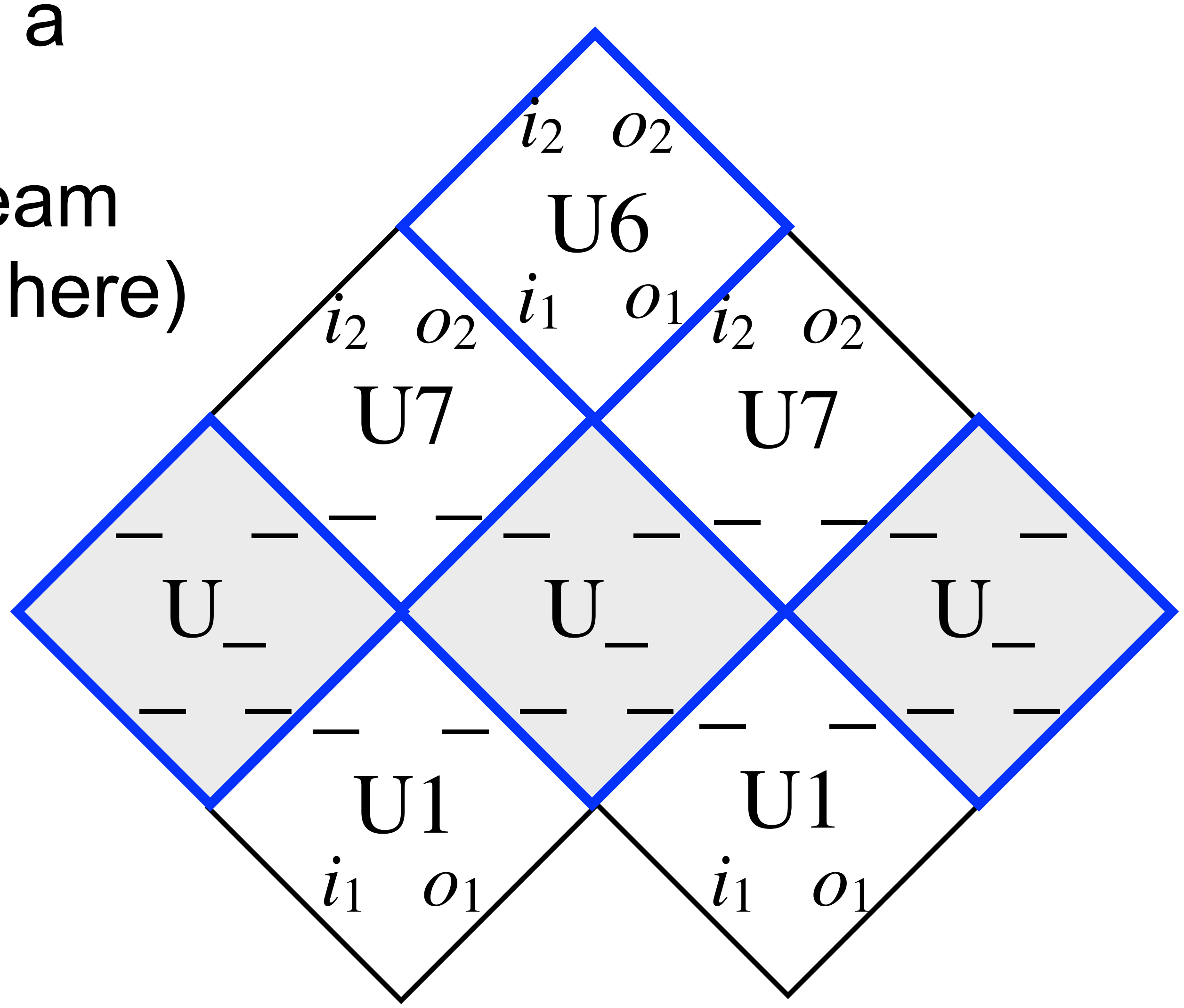


For each gate we have 4 tiles, 1 or which sticks

Glues encode rows

6-bit universal tileset: overview

2. Wrap into a tube along boundary/seam (“_” = no bit here)



2.1. U_ does not encode input/output bits. U_ encodes “boundary”

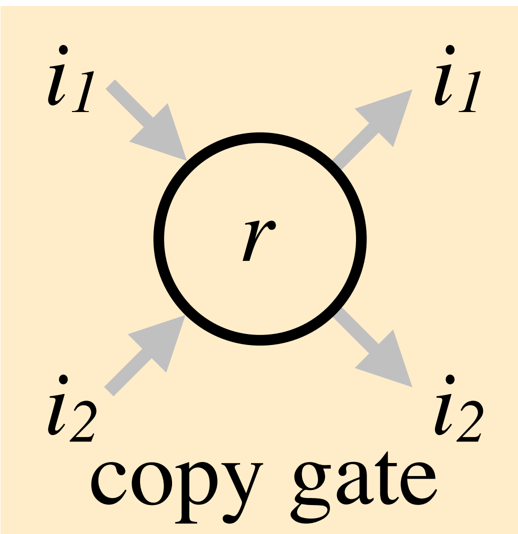
2.2. U2,…,U6 have 2 input and 2 output bits. U1 & U7 have only 1 input and 1 output bit.

3. Asynchronous update semantics: assembly frontier grows asynchronously rather than layer-by-layer (does not change expressivity of circuit versus tile model, roughly speaking)

But can we afford all those tiles?

From gates to tiles: savings

- Let's convert the set of *R*-bit universal gates into tiles, and examine at the resulting *R*-bit universal tile set
- Suppose I have two different gates, e.g. copying and sorting. If I convert each into 4 tiles I get 8 tiles, but lets look closer at some tile-savings:



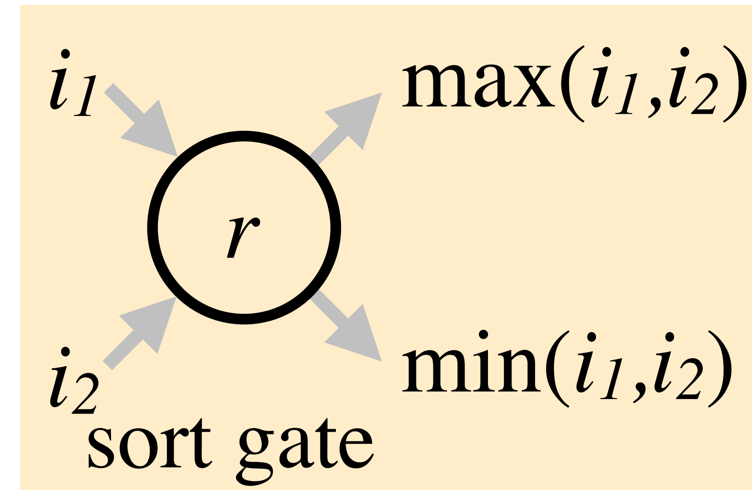
$i_1, i_2 \in \{0,1\}$

i1	i2	o1	o2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

copy gate

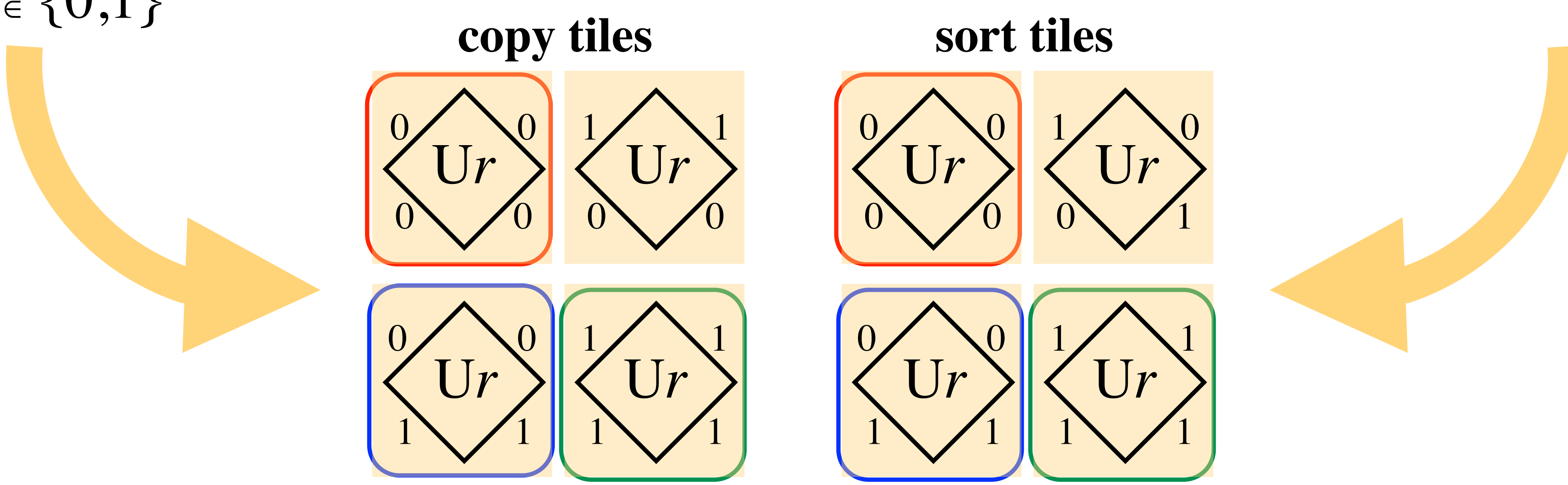
i1	i2	o1	o2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

sort gate



$i_1, i_2 \in \{0,1\}$

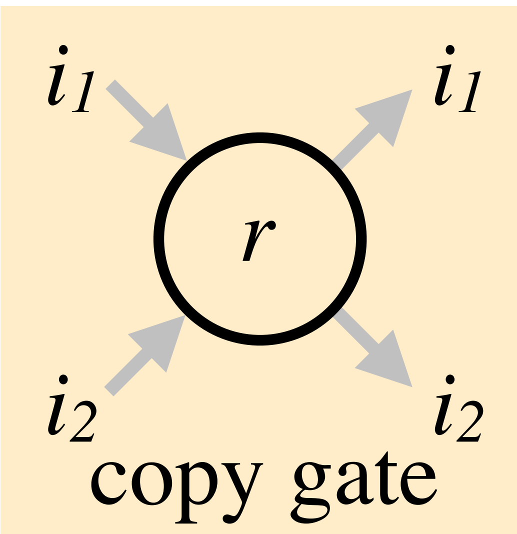
Truth tables: 3 identical row-pairs!



3 identical tile-pairs!

From gates to tiles: savings

- Let's convert the set of *R*-bit universal gates into tiles, and examine at the resulting *R*-bit universal tile set
- Suppose I have two different gates, e.g. copying and sorting. If I convert each into 4 tiles I get 8 tiles, but lets look closer at some tile-savings:



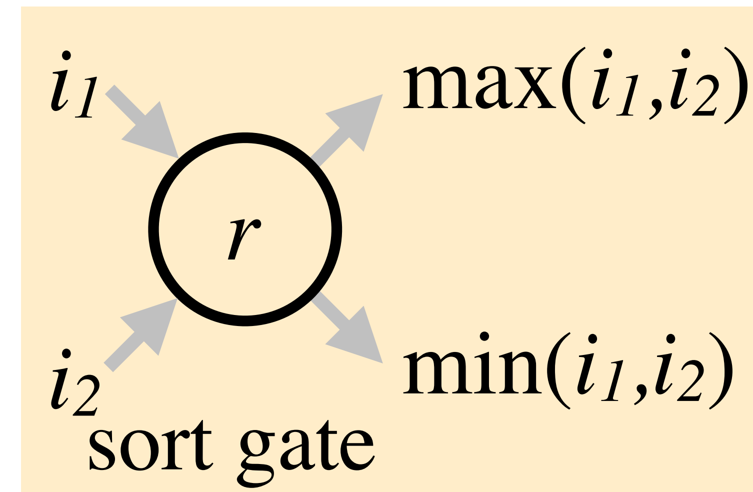
$i_1, i_2 \in \{0,1\}$

i1	i2	o1	o2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

copy gate

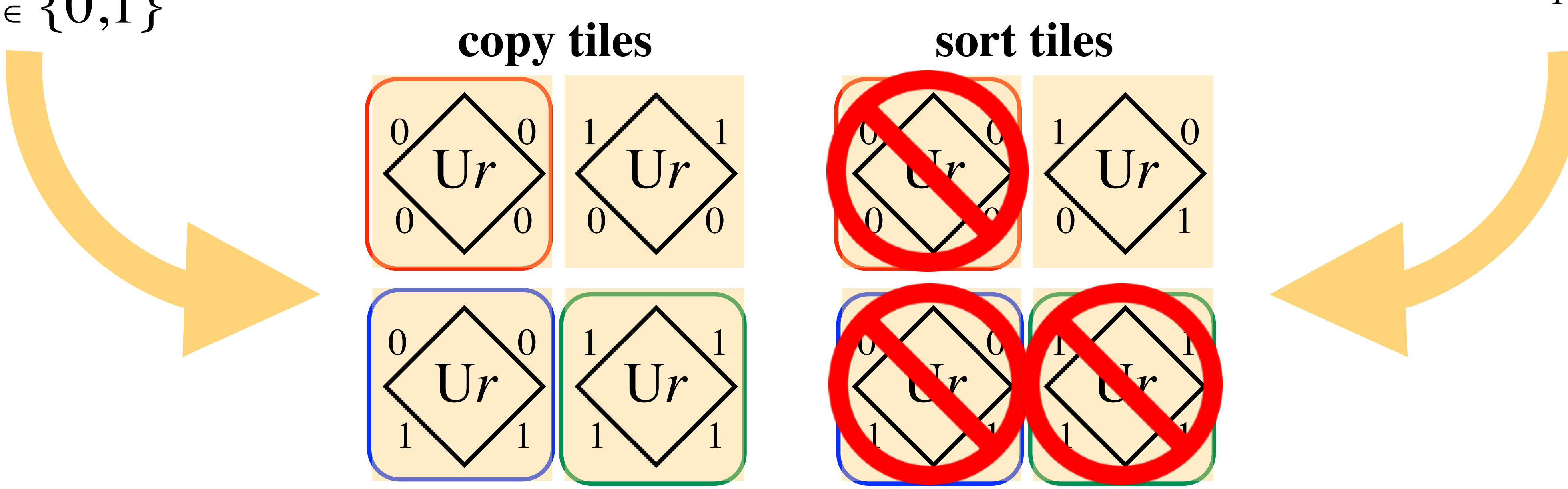
i1	i2	o1	o2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

sort gate



$i_1, i_2 \in \{0,1\}$

Truth tables: 3 identical row-pairs!



3 identical tile-pairs!

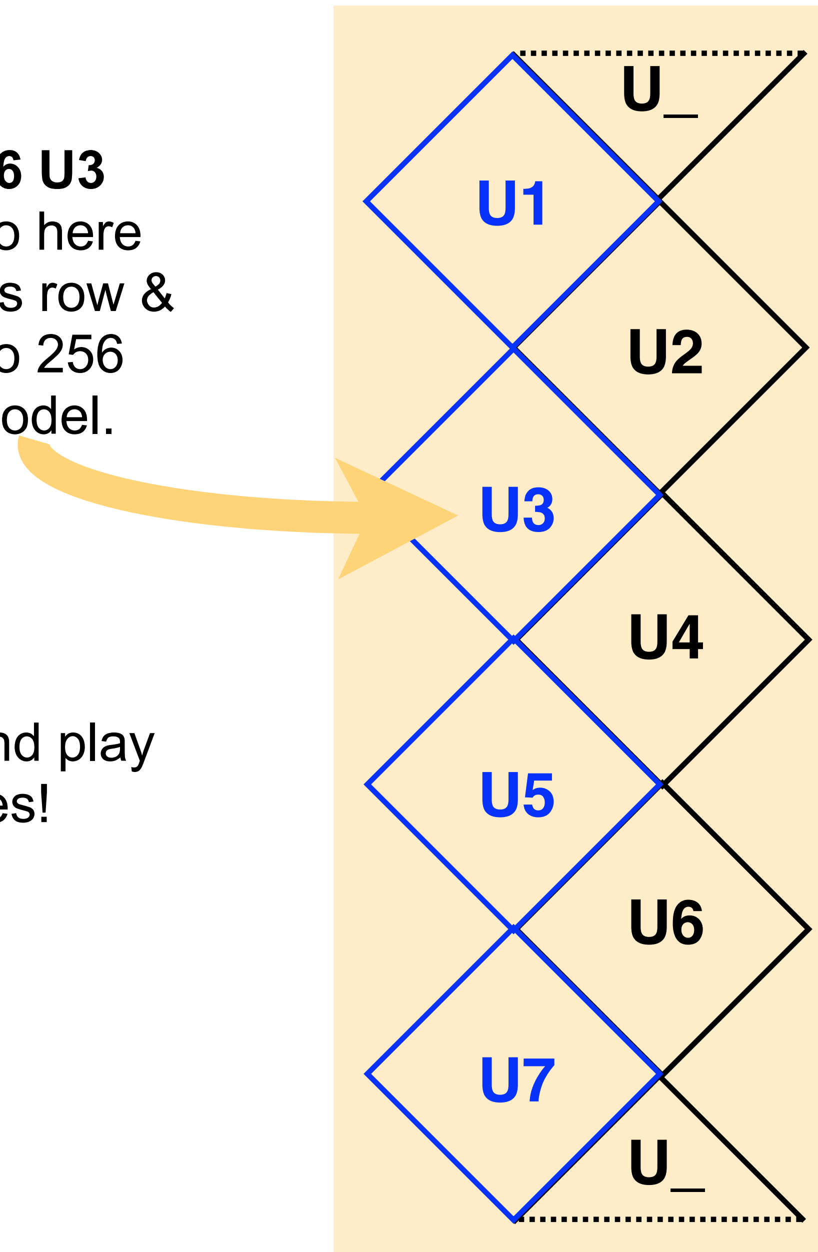
Only 5 tile types needed to do both copying and sorting!

6-bit universal tileset: overview

- Intuition from previous slide: Tiles separate the 4 “elementary operations” of a gate into 4 individual tiles, which results in **fewer tile types** in our universal tile set than gates in the universal gate set
- So how many tiles in the *R*-bit universal tile set?

E.g. U4: There are **16 U3 tile types** that can go here (a tile is defined by its row & 4 bits), as opposed to 256 gates in the circuit model.

The user may plug and play with these 16 tile types!



4 U1 has 1 input and 1 output bit

16

16

16

16

16

U2,3,4,5,6 each have 2 input and 2 output bits, hence 16 tile types each

4 U7 has 1 input and 1 output bit

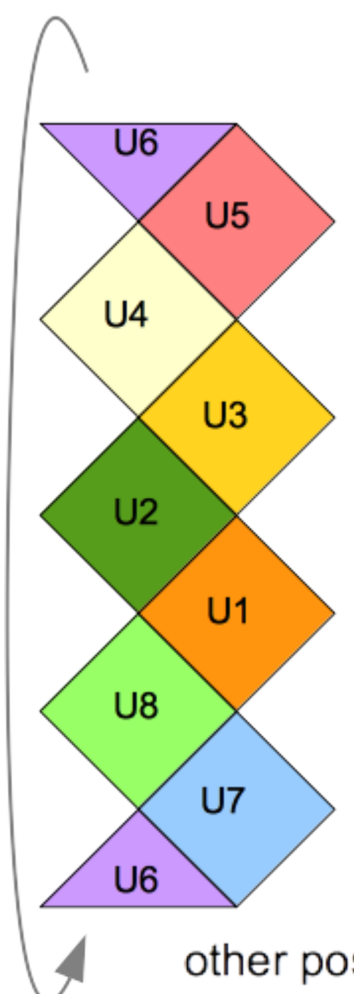
1 U_ is seam tile (represents no bits)

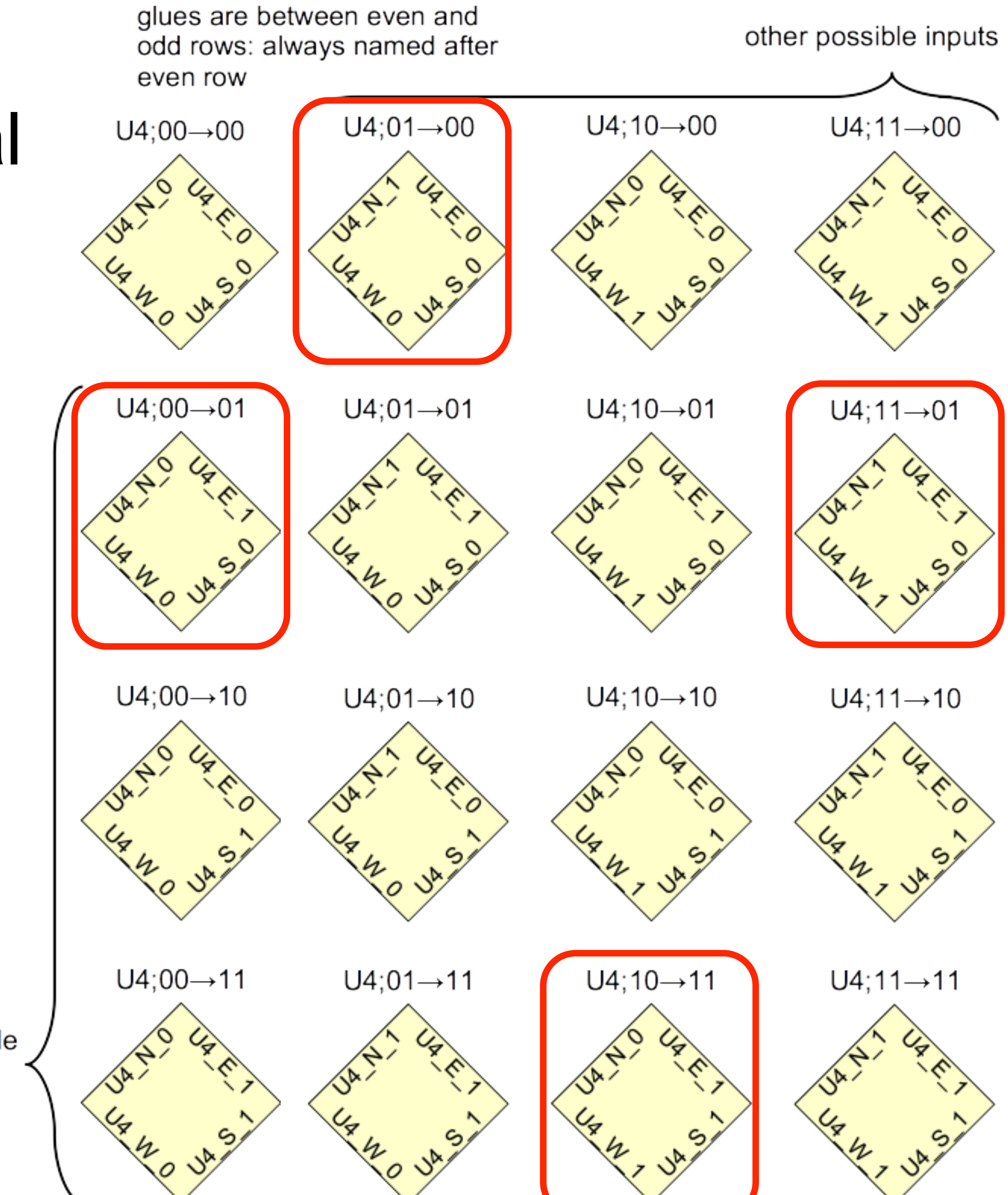
Total: 89 tile types

1,288 for 2-output gates.
168 for 1 output gates

6-bit universal tileset: details

8 rows U1–U8; each has disjoint subset of tile types





to compute a function, e.g.:

i1	i2	o1	o2
0	0	0	1
0	1	0	0
1	0	1	1
1	1	0	1

select one tile **from each column**

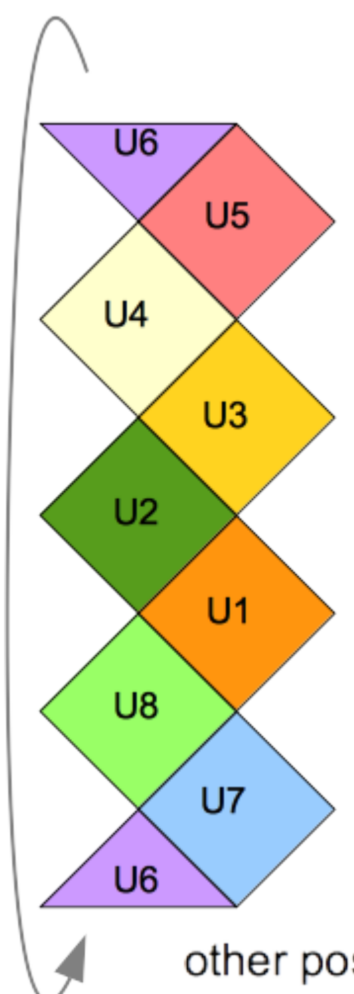
pic by Dave Doty

6-bit universal tileset: details

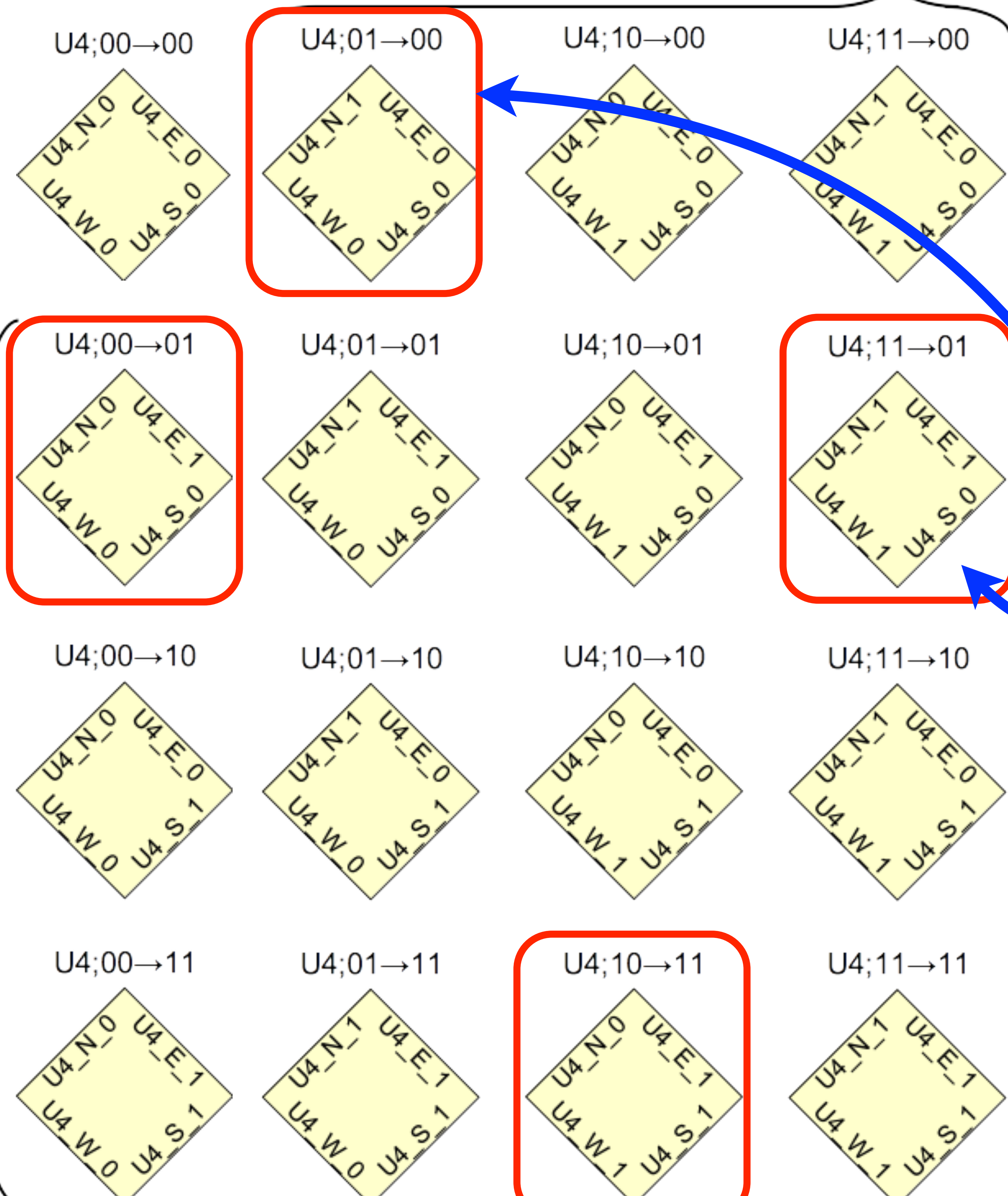
glues are between even and odd rows: always named after even row

other possible inputs

8 rows U1–U8; each has disjoint subset of tile types



other possible outputs on input 00



to compute a function, e.g.:

i1	i2	o1	o2
0	0	0	1
0	1	0	0
1	0	1	1
1	1	0	1

select one tile **from each column**

tiles in rows U3,U5,U6,U7 selected similarly

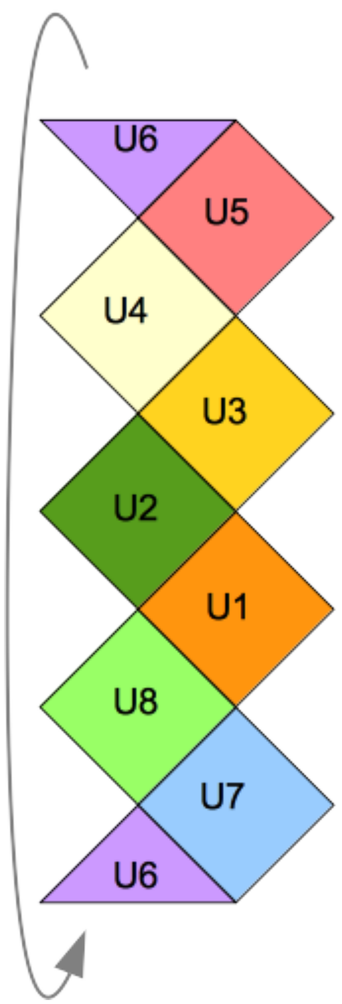
each computes a function $f : \{0,1\}^2 \rightarrow \{0,1\}^2$

pic by Dave Doty

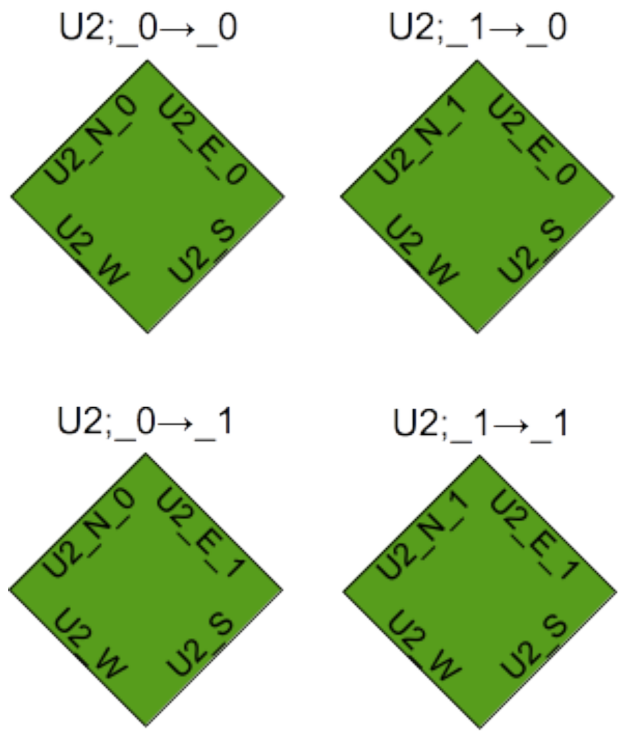
6-bit universal tileset: details

Special cases for rows near seam

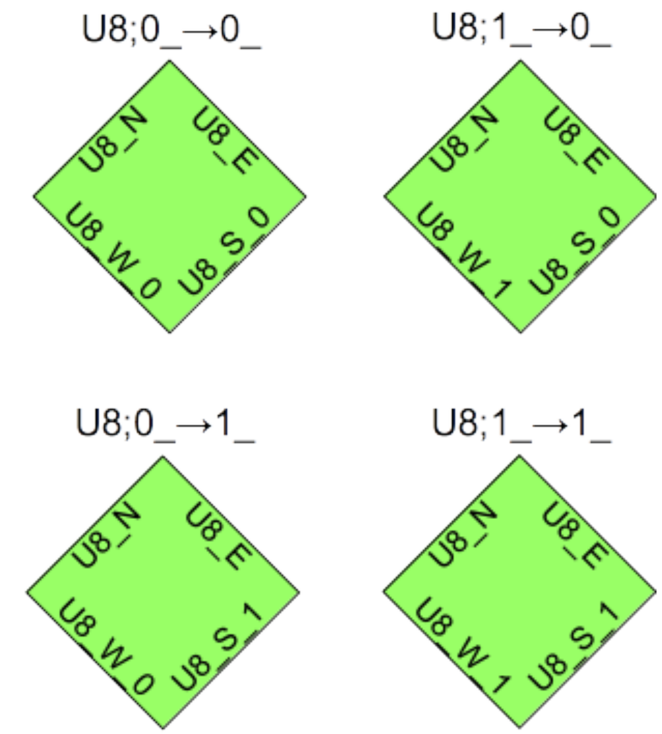
8 rows U1–U8; each has disjoint subset of tile types

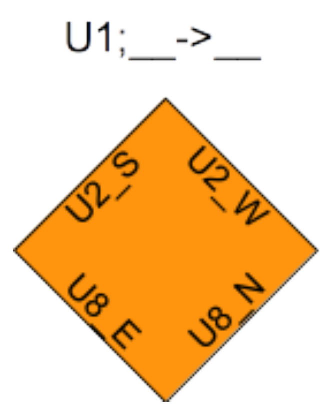


pic by Dave Doty



U2 and U8 have no bit on the helix they share with U1, so they compute a function $f : \{0,1\} \to \{0,1\}$



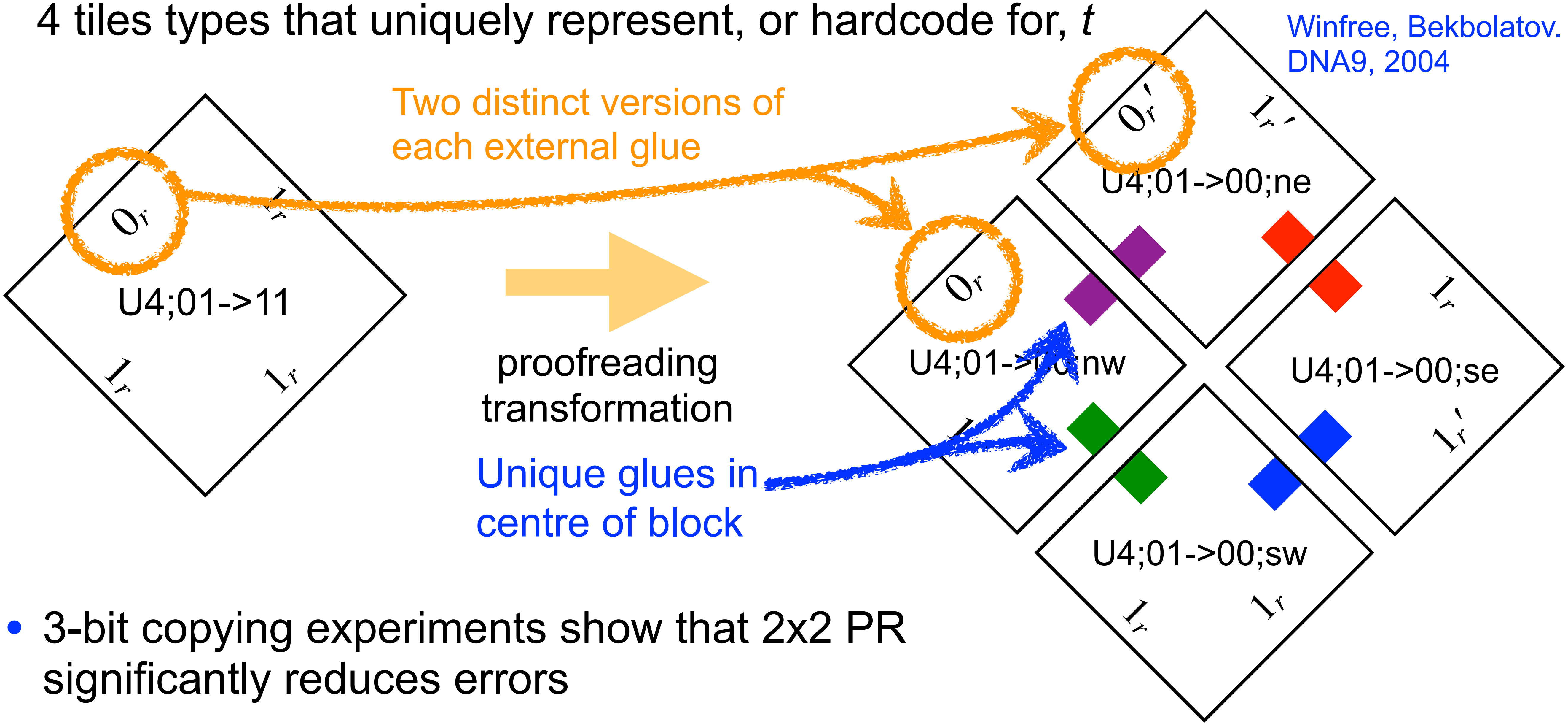


only 1 tile type on position U1, computes trivial function f:(_,_) → (_,_)

6-bit universal proofreading (PR) tileset

- Linear/polynomial redundancy for exponential error reduction
- 2x2 PR transformation: each tile type *t* is transformed into a 2x2 block of 4 tiles types that uniquely represent, or hardcode for, *t*

Winfree, Bekbolatov. DNA9, 2004

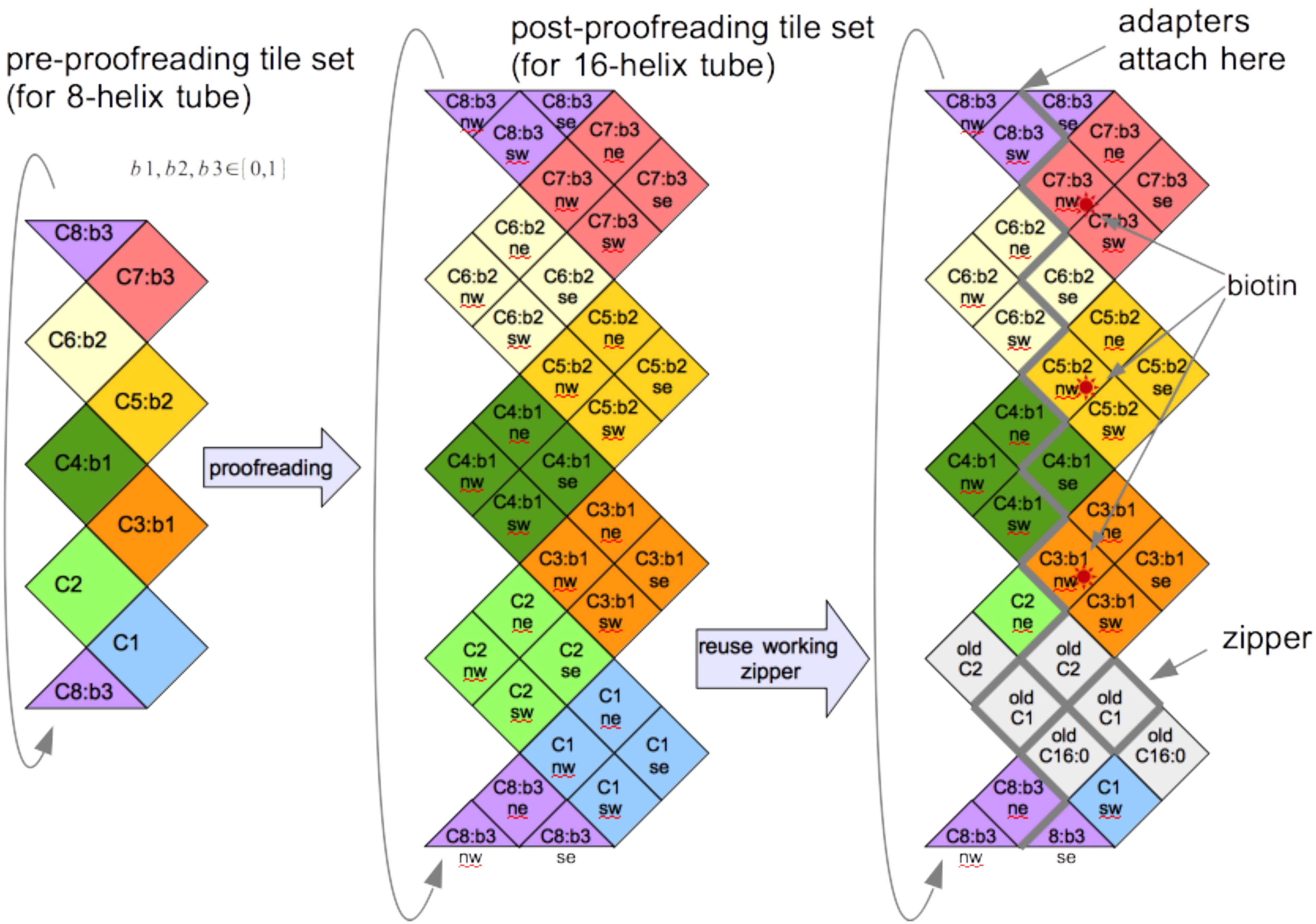


- 3-bit copying experiments show that 2x2 PR significantly reduces errors
- Transforms 89 tiles into **356 proofreading tiles**
- Caveat: we will use only a single tile type along the seam (hence, the 2x2 “U_” block at the seam is not a proofreading block). => 4*89-1=**355 unique strands**

Key property: 1 error forces a 2nd error in the same block, squaring the error rate

3-bit proofreading copying tileset

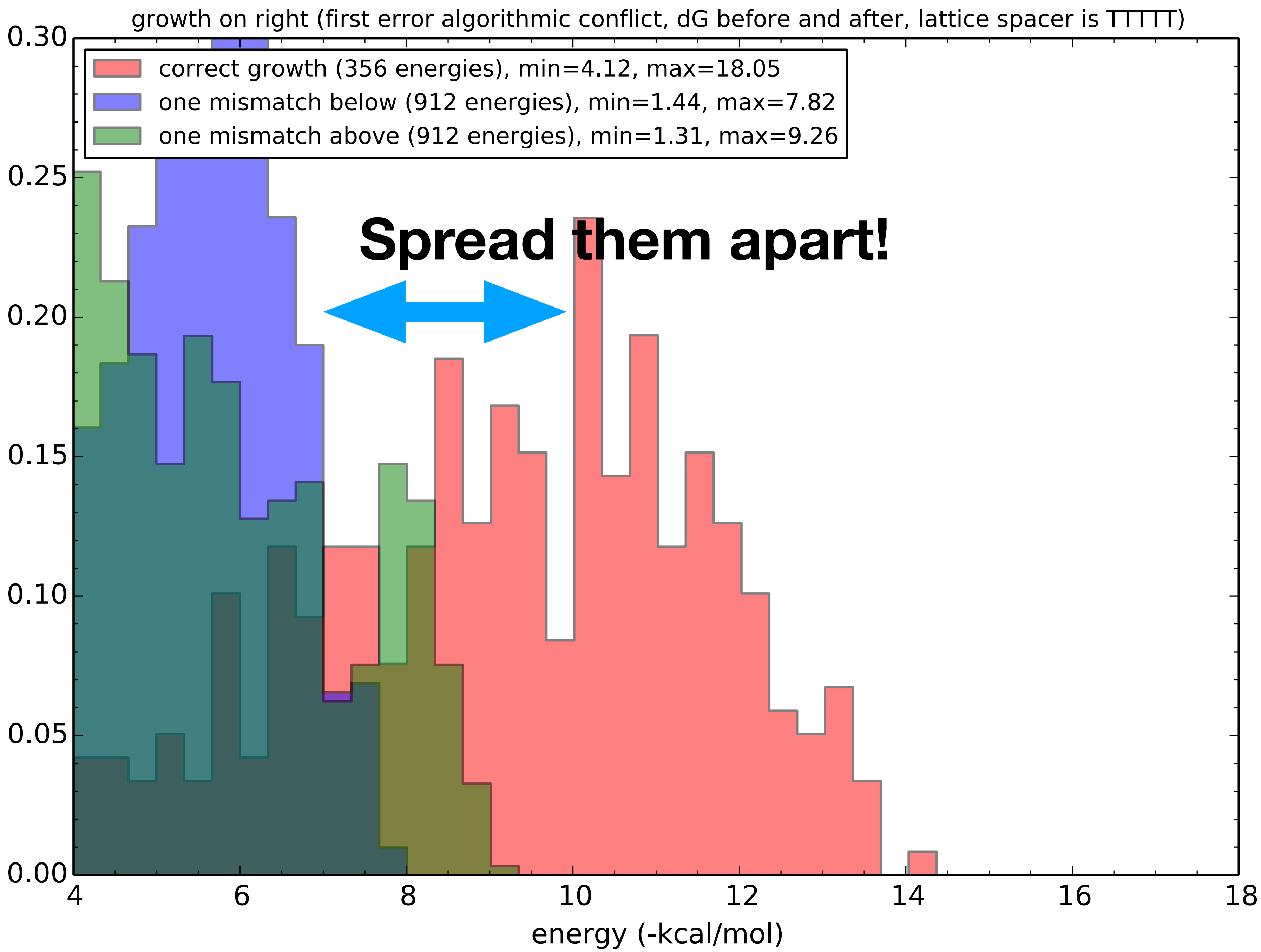
- To give an idea of what a 2x2 proof-reading transformation is here is a 3-bit proofreading copying applied to the 3-bit copying tile set (i.e. for a different tile set)



pic by Dave Doty

Sequence design

Random sequences will not work

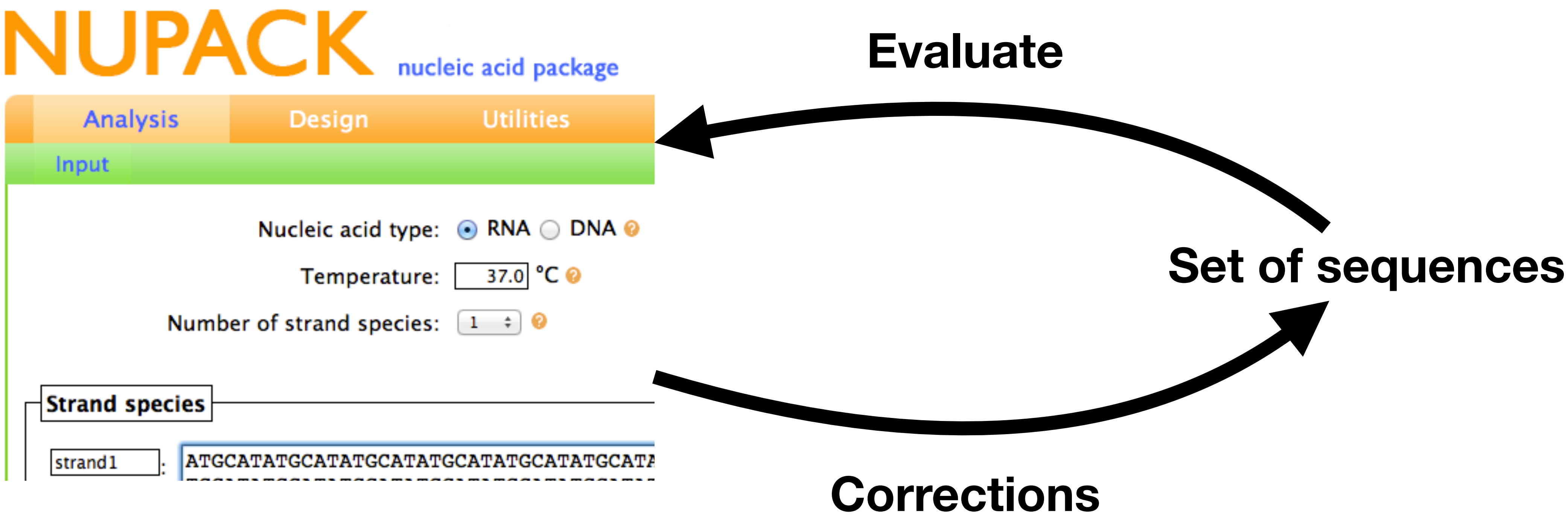


Random sequences over 3-letter code with 1 base exception, and domain-pairs ending with AT stack

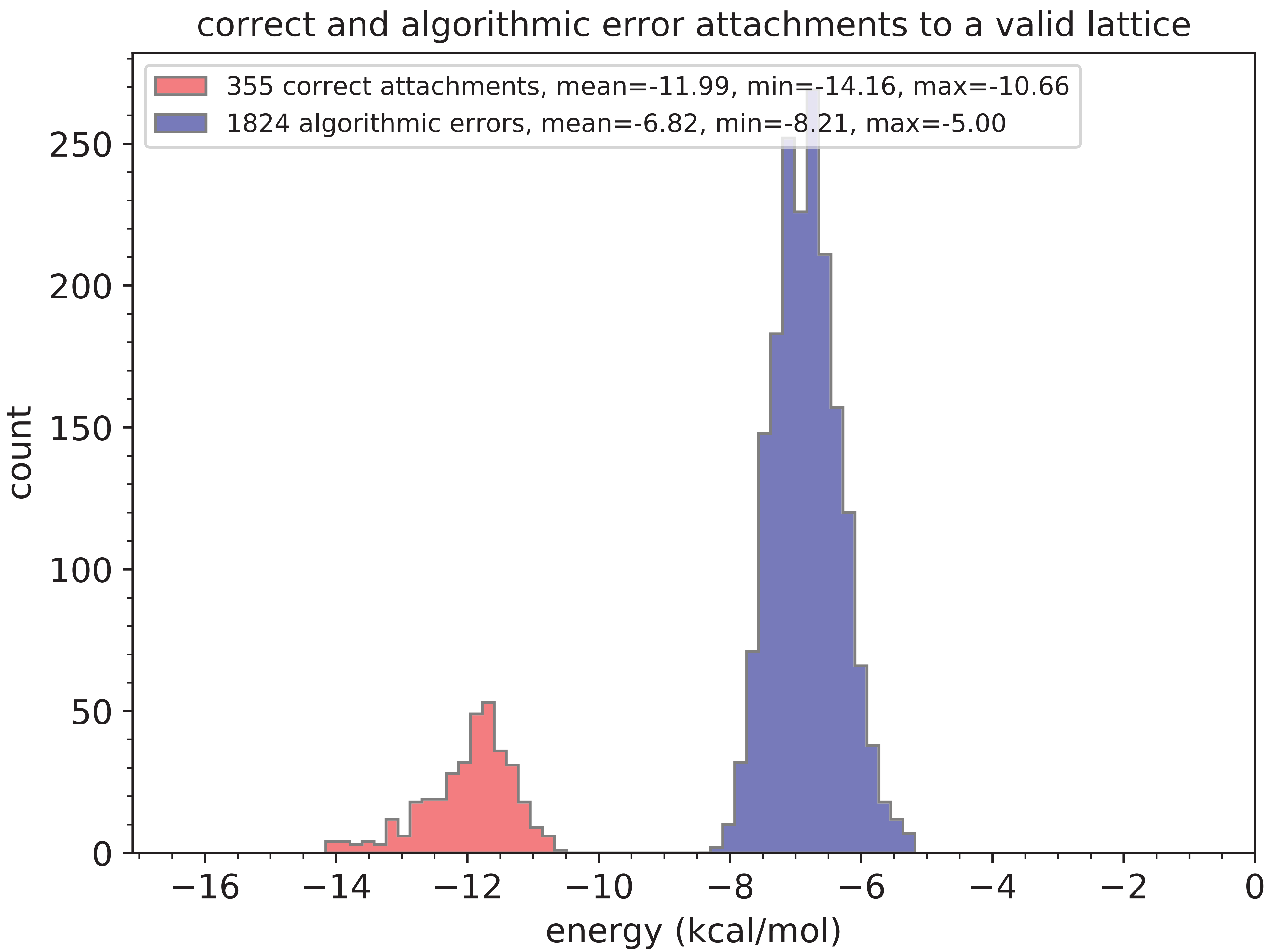
What do we want?

1. No "self-folding"
2. Clean lattice boundary
3. Minimize interactions between strand pairs
4. Uniform correct binding: in a tight range
5. Incorrect binding should have a much higher energy

An iterative process



Designed sequences



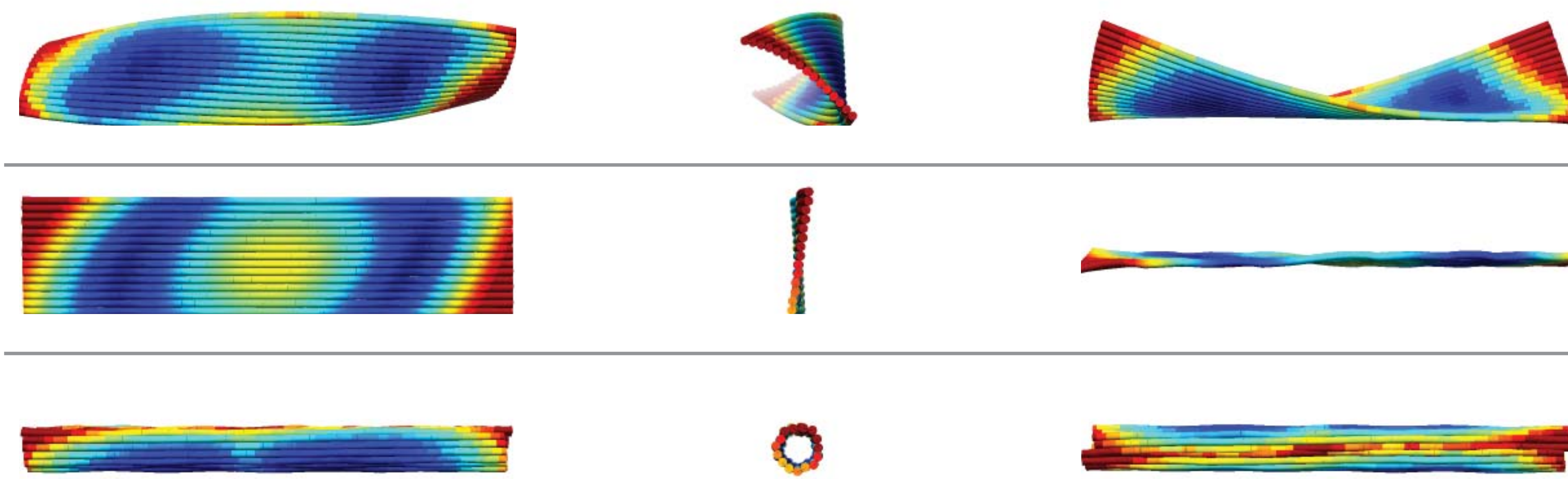
The experiments

The seed: a DNA origami

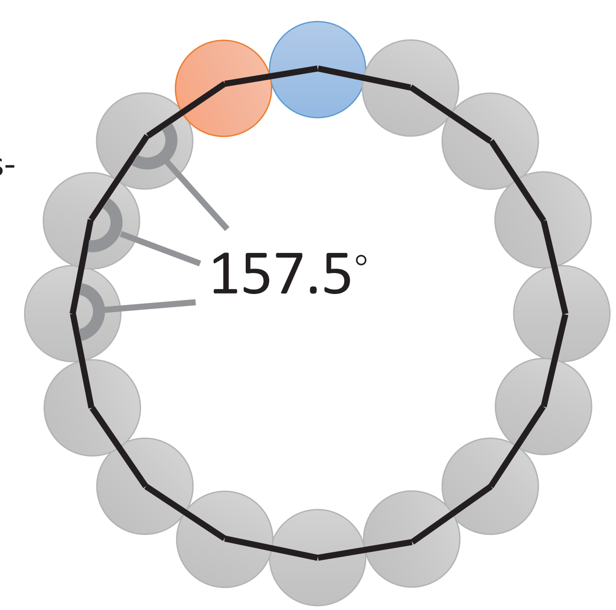
Classic rectangle

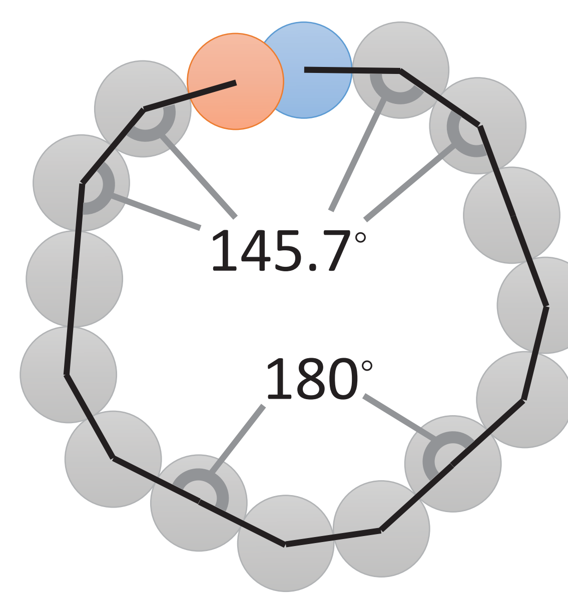
Twist correction

Barrel correction



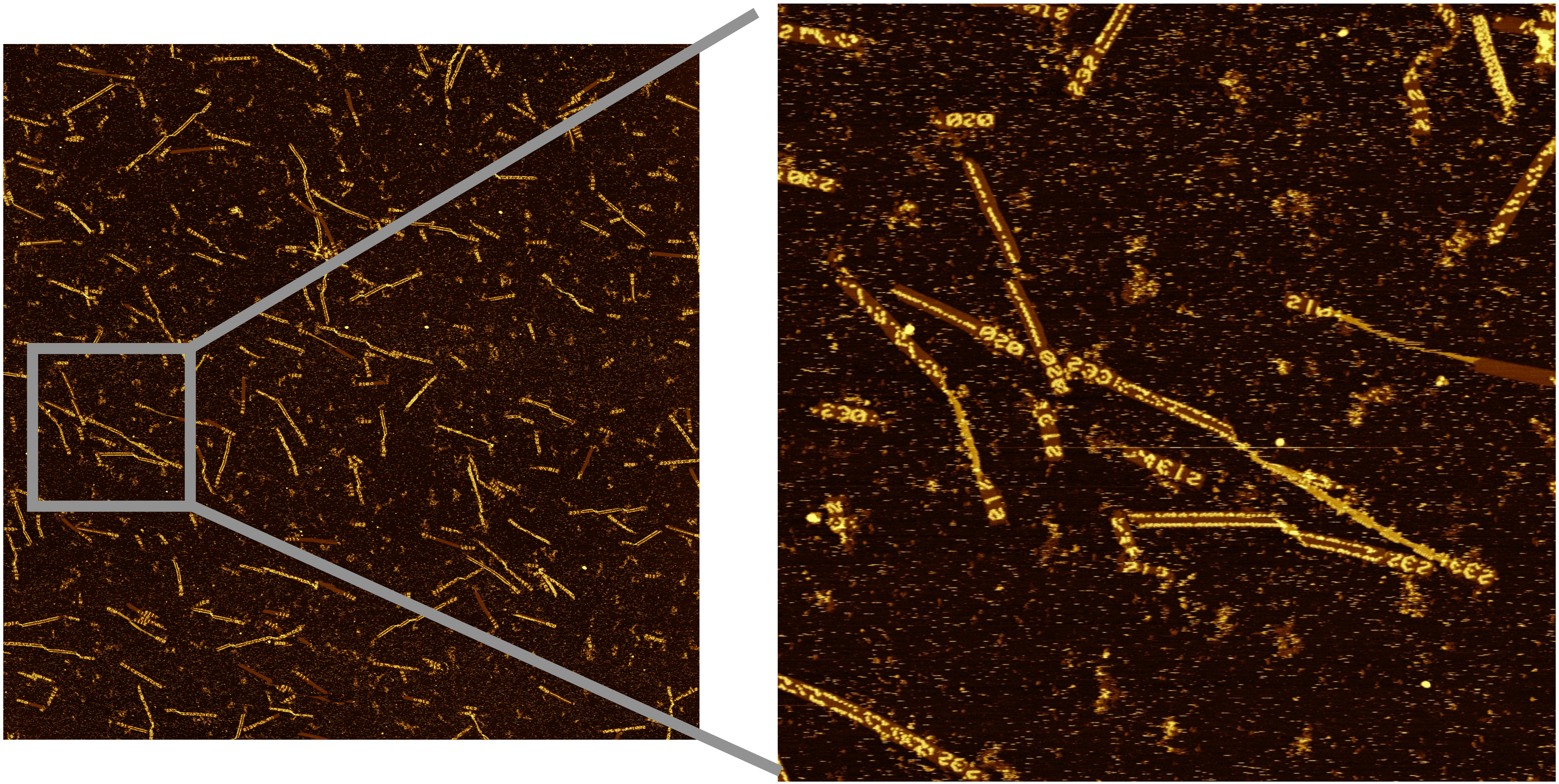
tiles: idealized cross-section of 16-helix nanotube of single-stranded tiles with crossover between all adjacent helices: regular 16-gon





seed: idealized cross-section of 16-helix DNA origami barrel without crossover from top to bottom helix: irregular 16-gon

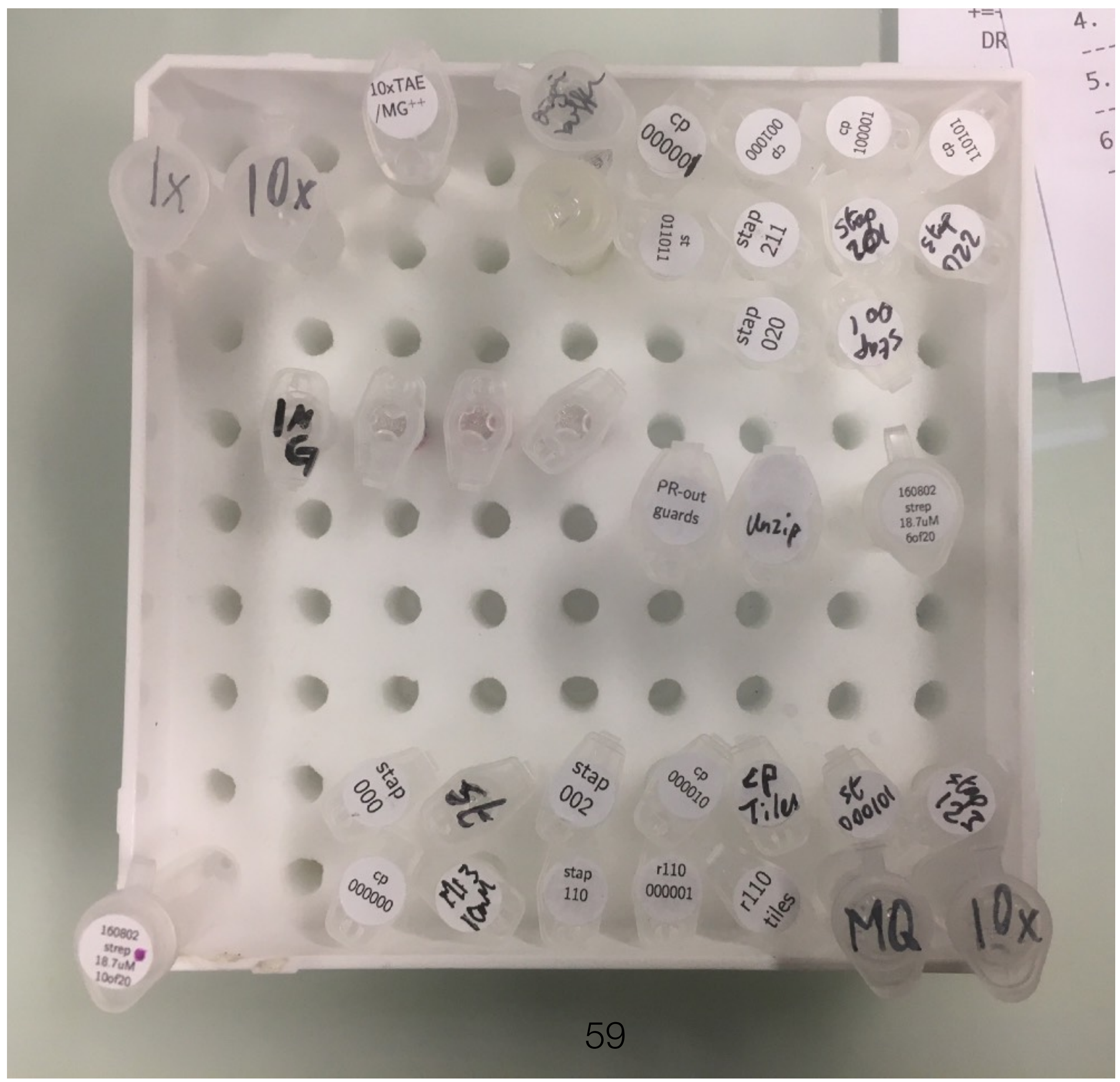
Barcode



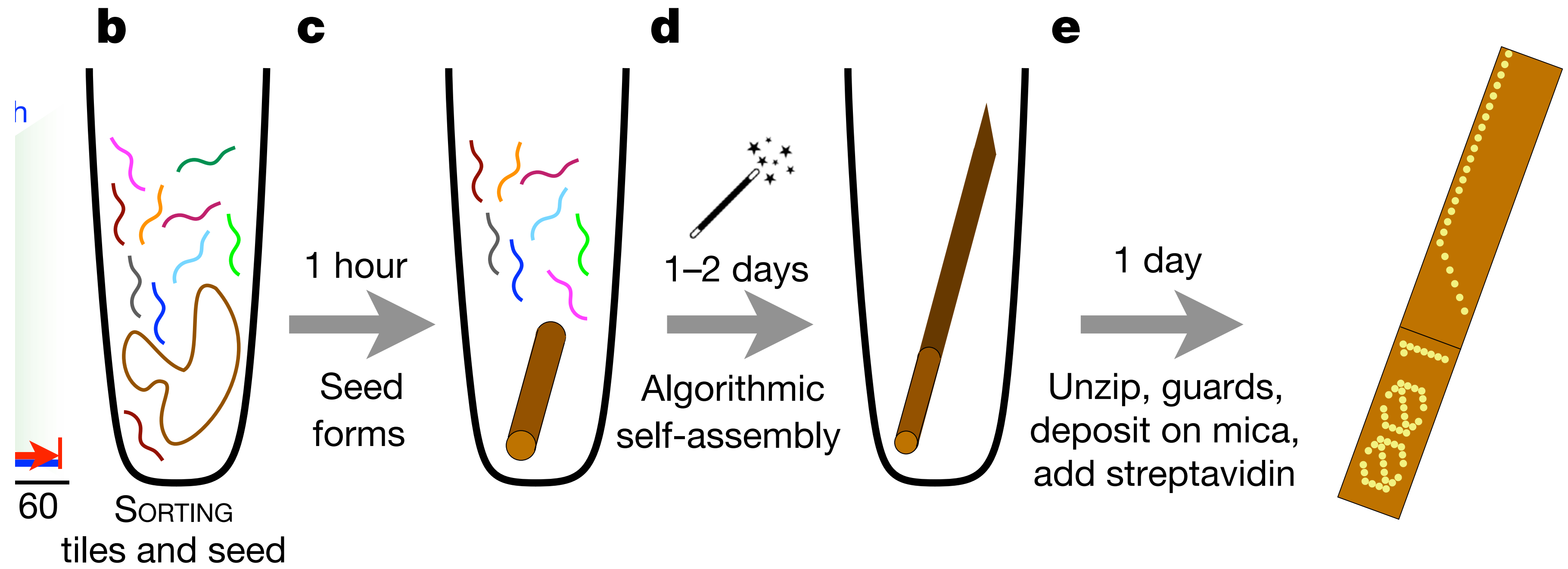
Seed barcodes allow to image many circuits/inputs at the same time

Preparing the tiles

- Mix of the tile strands for each of the circuits in an individual properly labelled tube



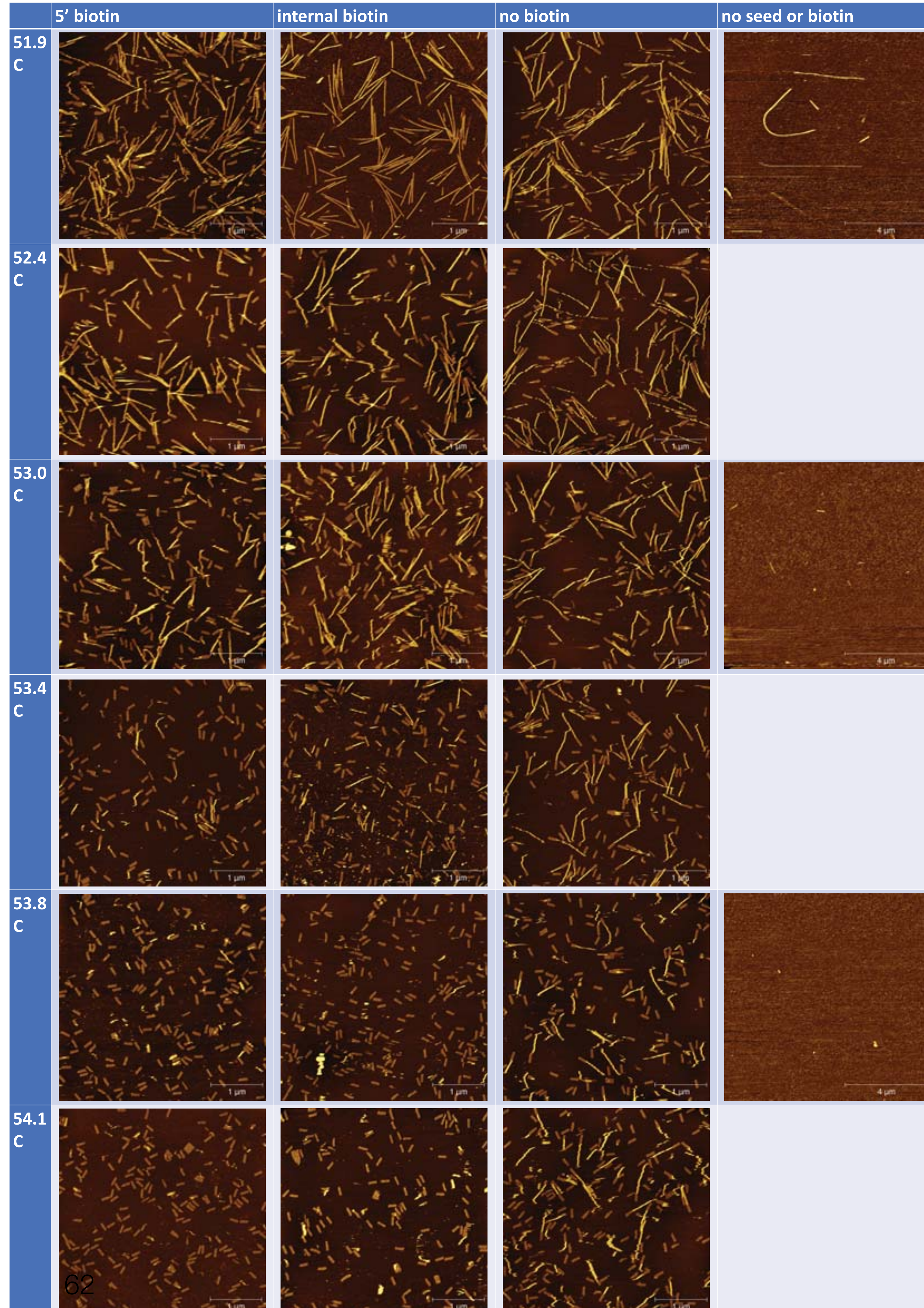
Protocol



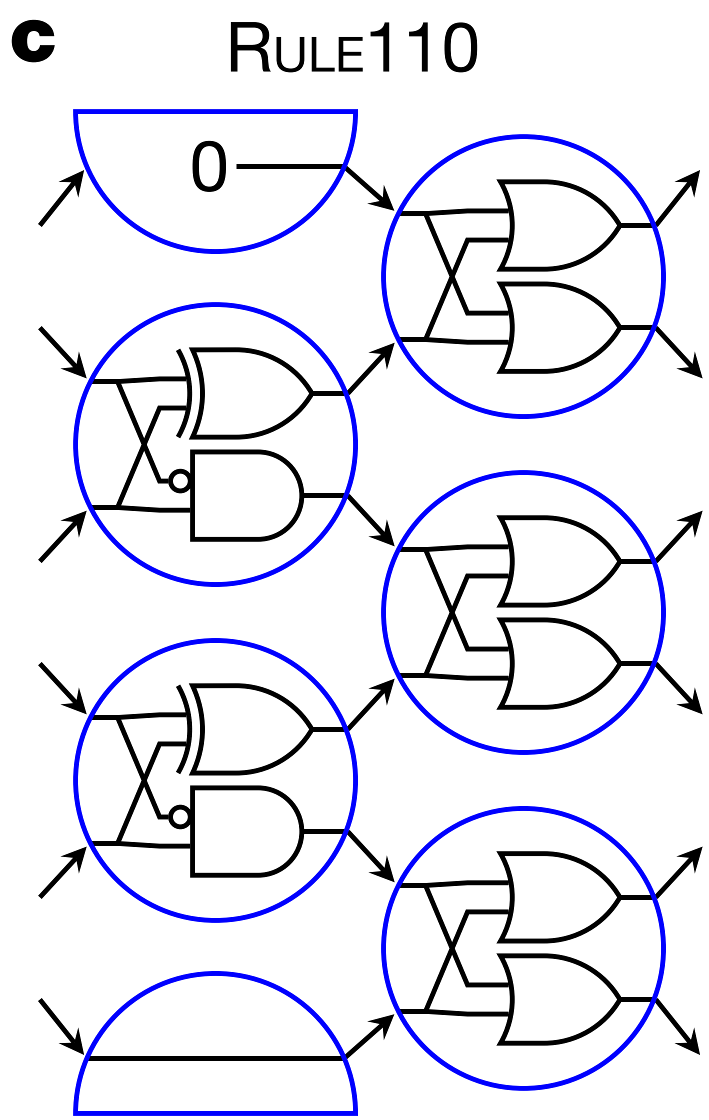
1. **Origami**
   1.1. Mix scaffold and staples and adapters
   1.2. Heat at 90°C and let it cold down to 51.8°C slowly (1h)
2. **Growth**
   2.1. Add tiles
   2.2. Let it grow at 51.8°C for 1 day
3. **Guards**
   3.1. Add Guard staples
   3.2. Let it attach for 4h
4. **Unzip**
   4.1. Add the unzipers
   4.2. Let it rest for 1 night
5. **Cool down to room temperature and Image!**

The result

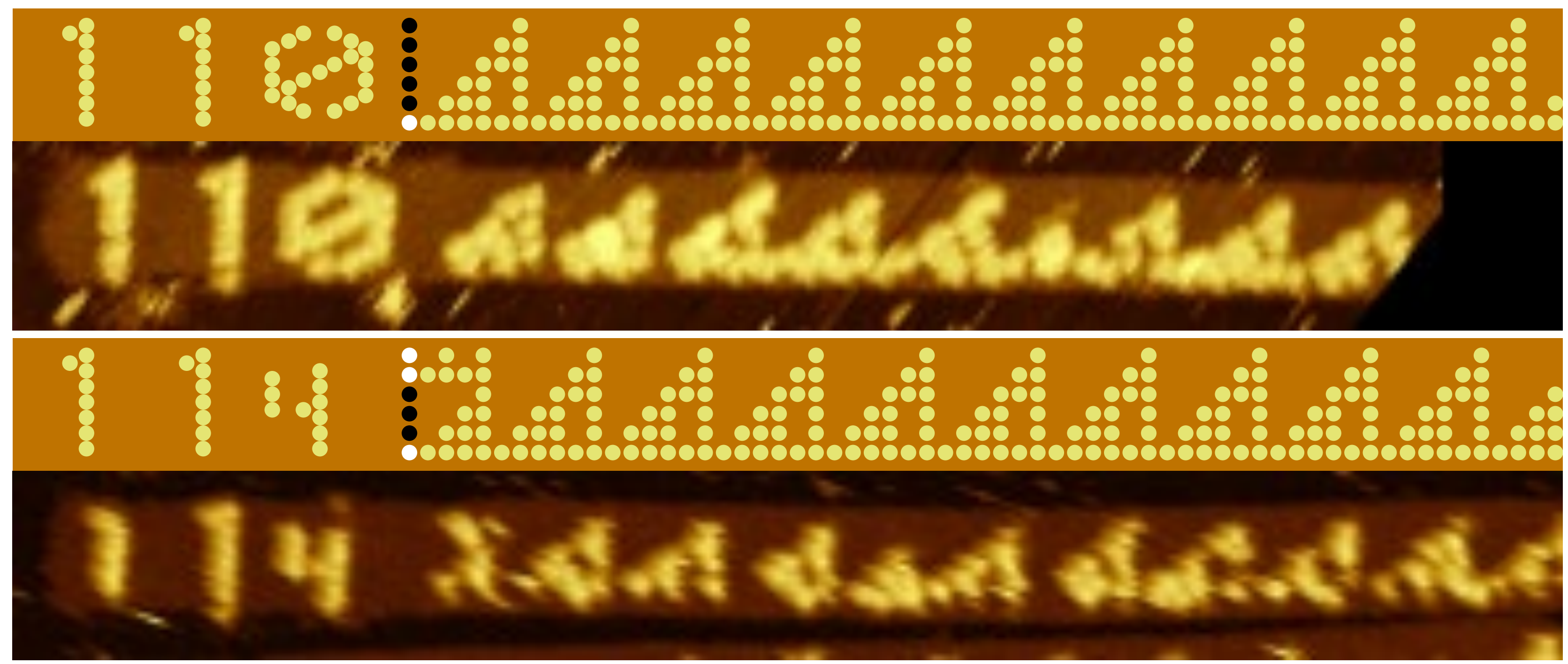
Influence of Temperature



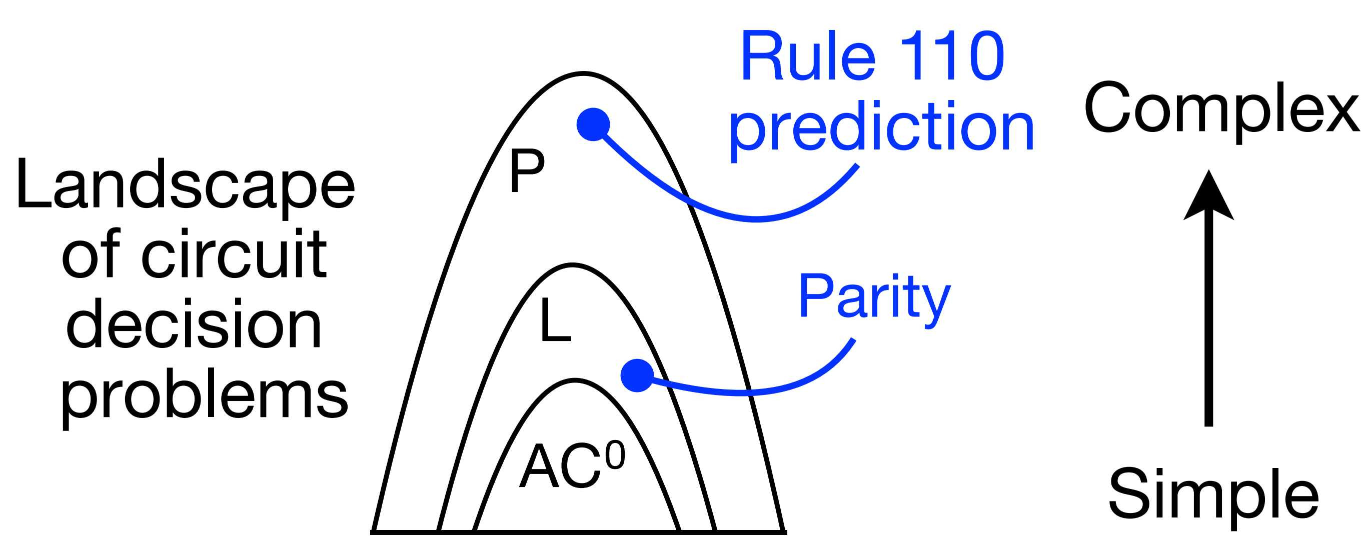
Rule 110: Turing complete!



Simulation of a cellular automaton



Tile-attachment error rate 0.03%±0.009
Number of tiles attached 48,789

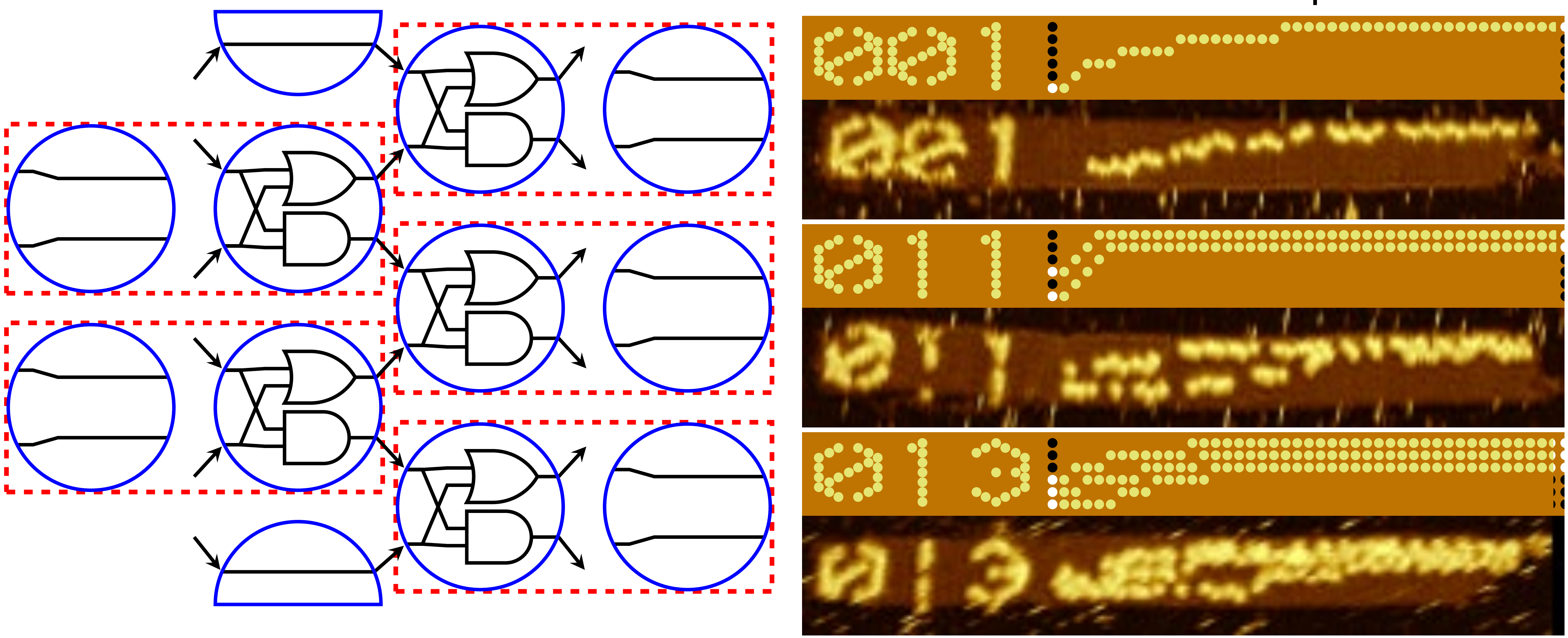


Lazy sorting

I

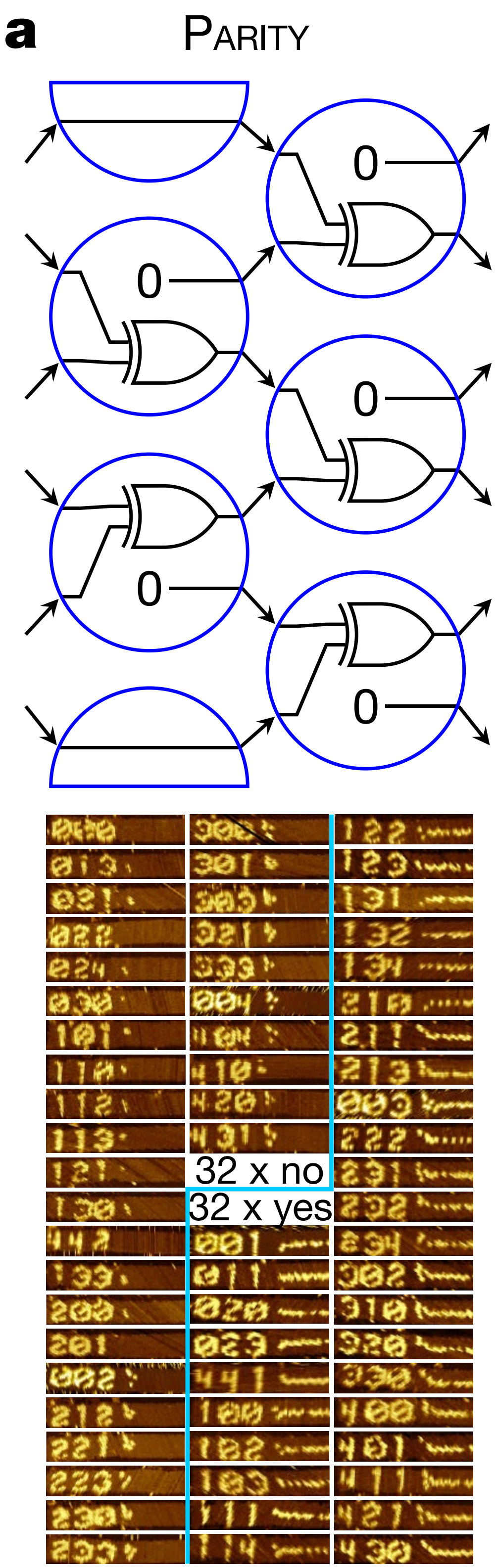
LazySorting

Sort 1s to the top



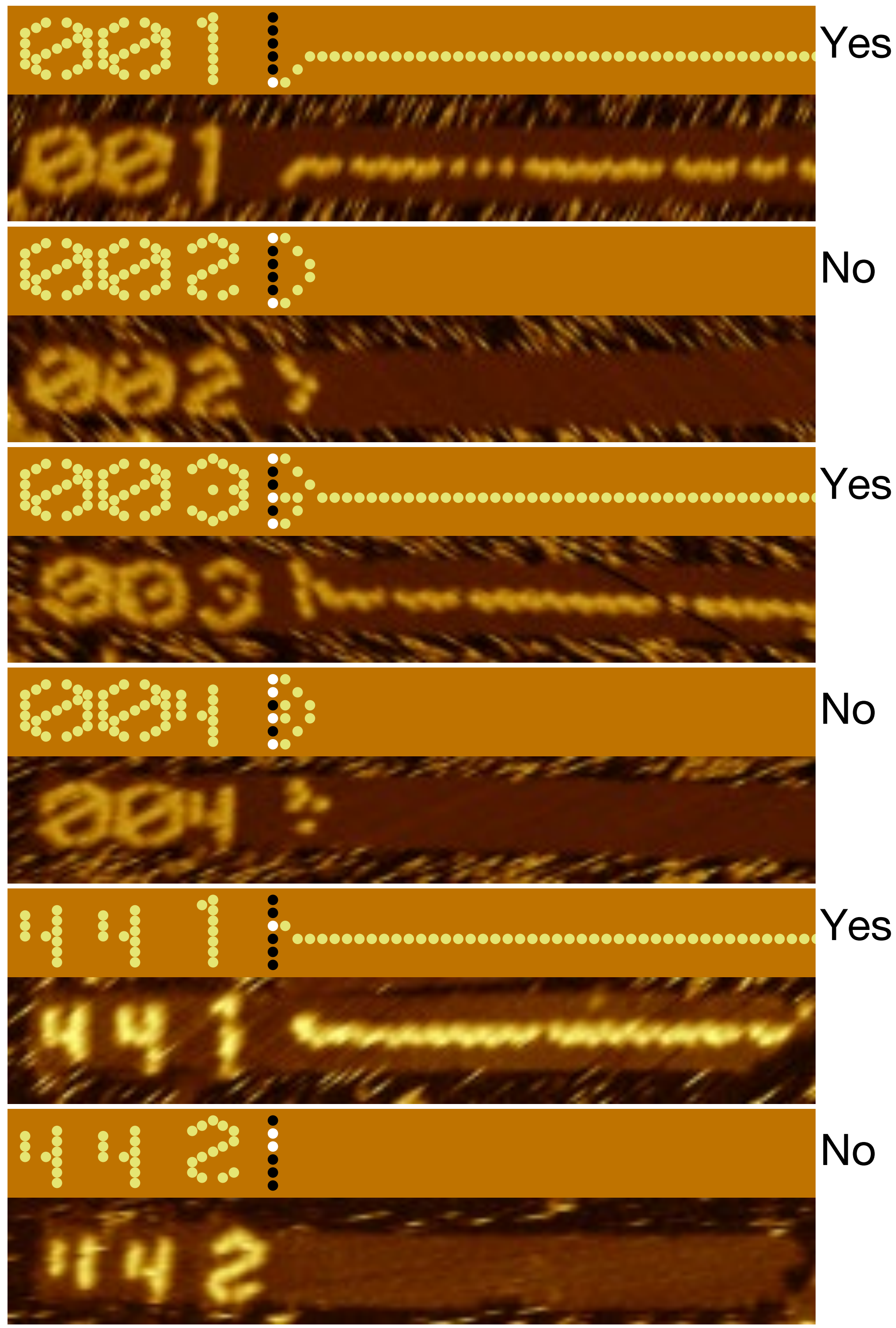
Tile-attachment error rate 0.03%±0.005

Parity

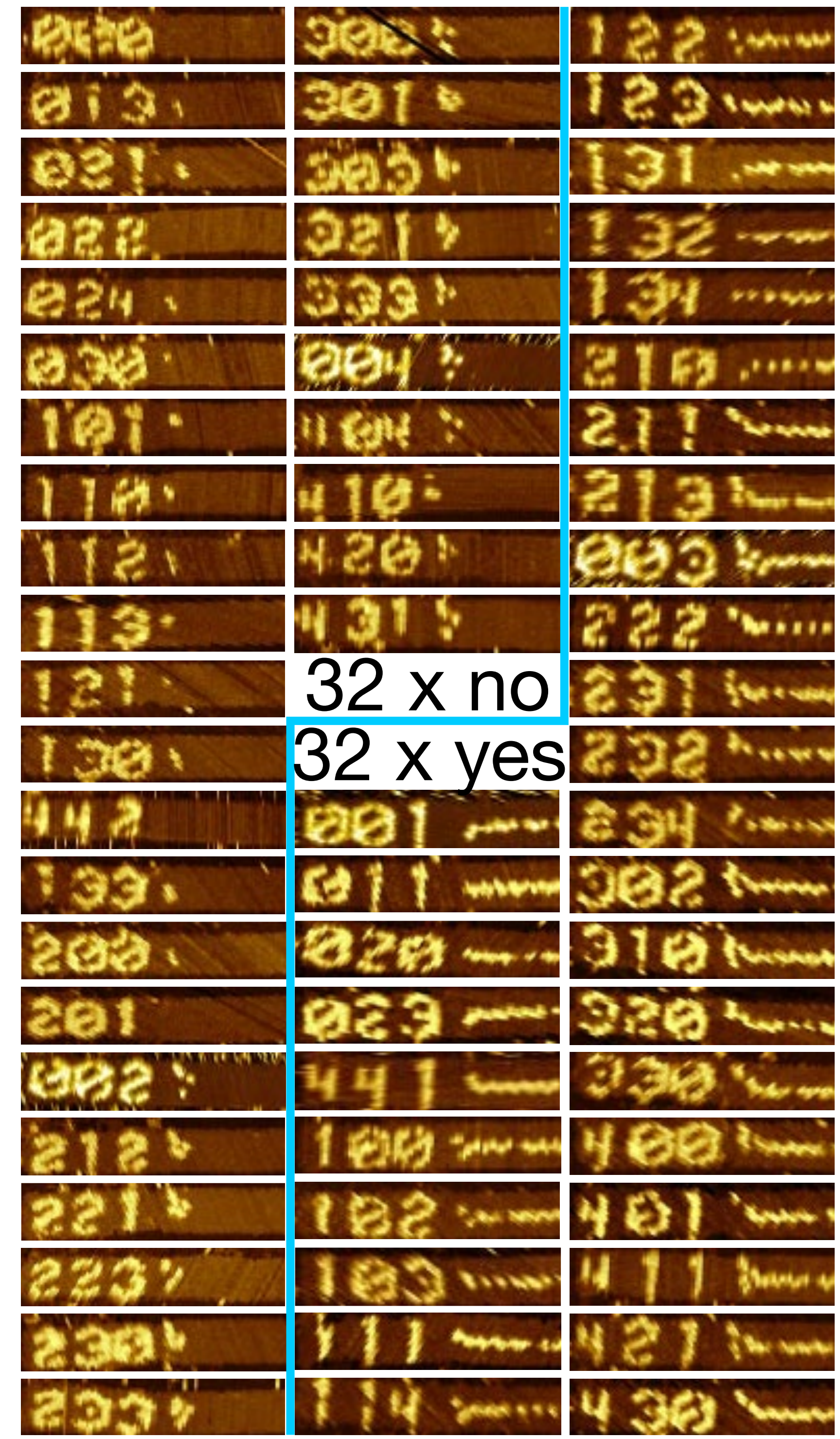


$2^6 = 64$ inputs

Is the number of 1s odd?



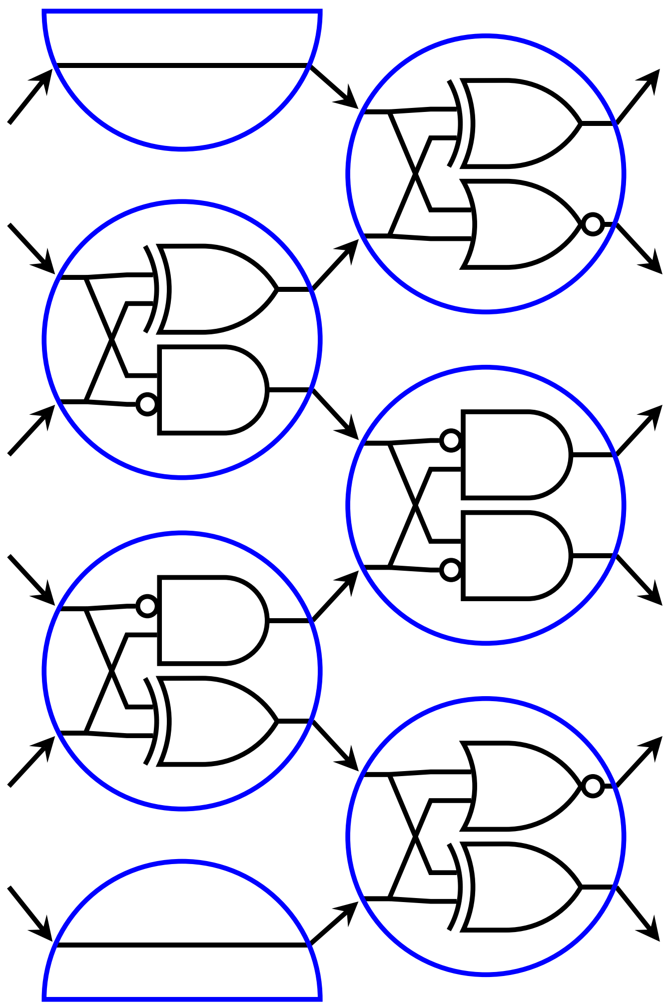
Tile-attachment error rate 0.03%±0.001
Number of tiles attached 1,318,163



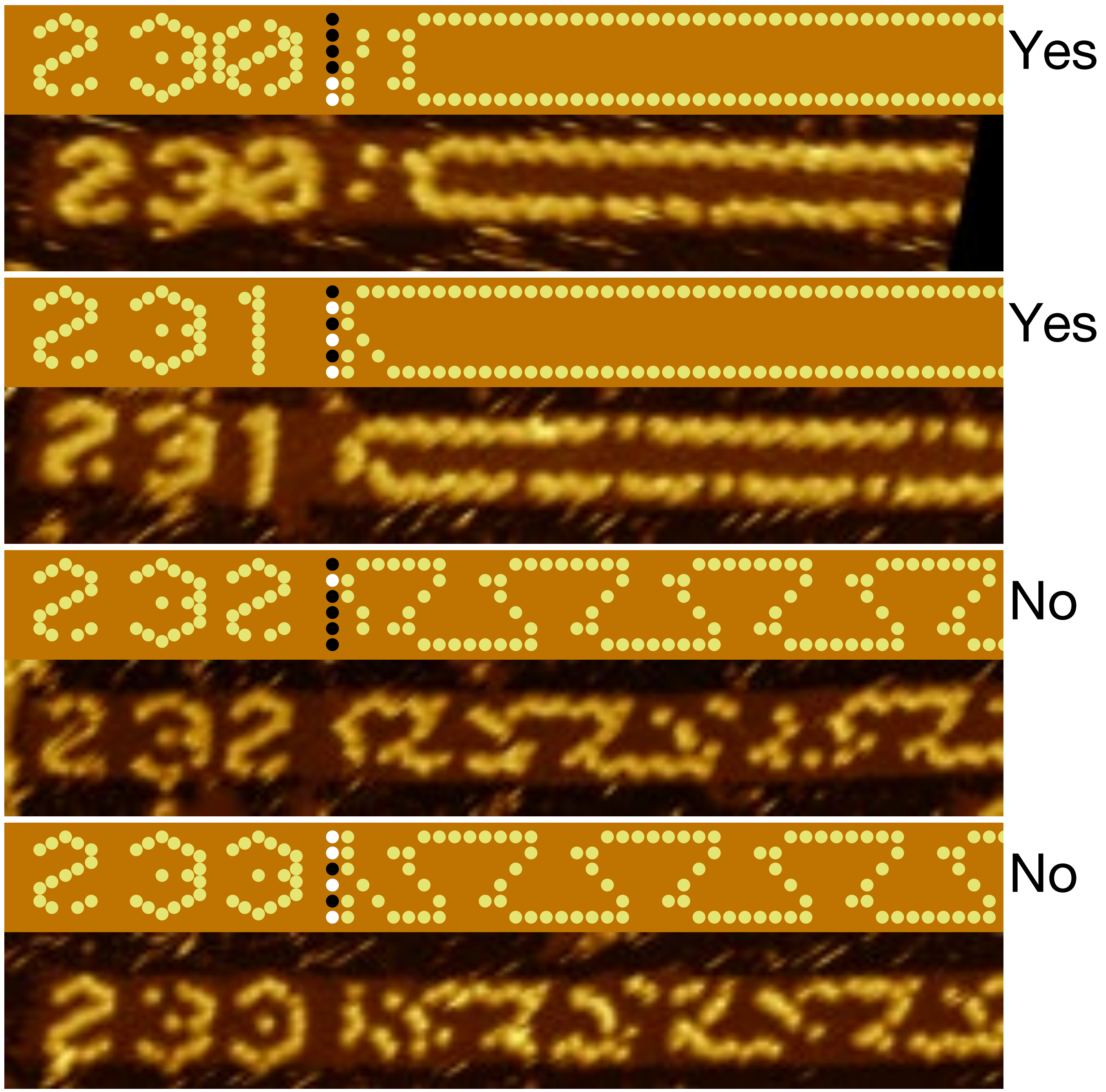
$2^6 = 64$ inputs

Multiple of 3?

b

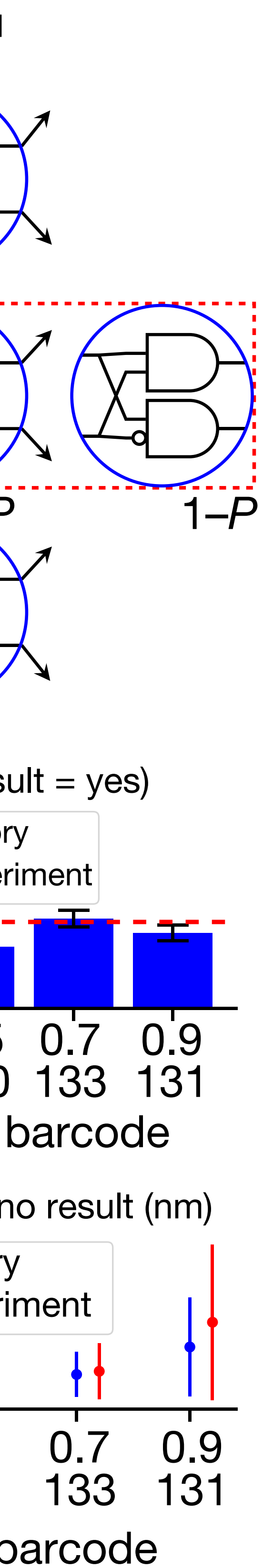


Is the input binary number a multiple of 3?

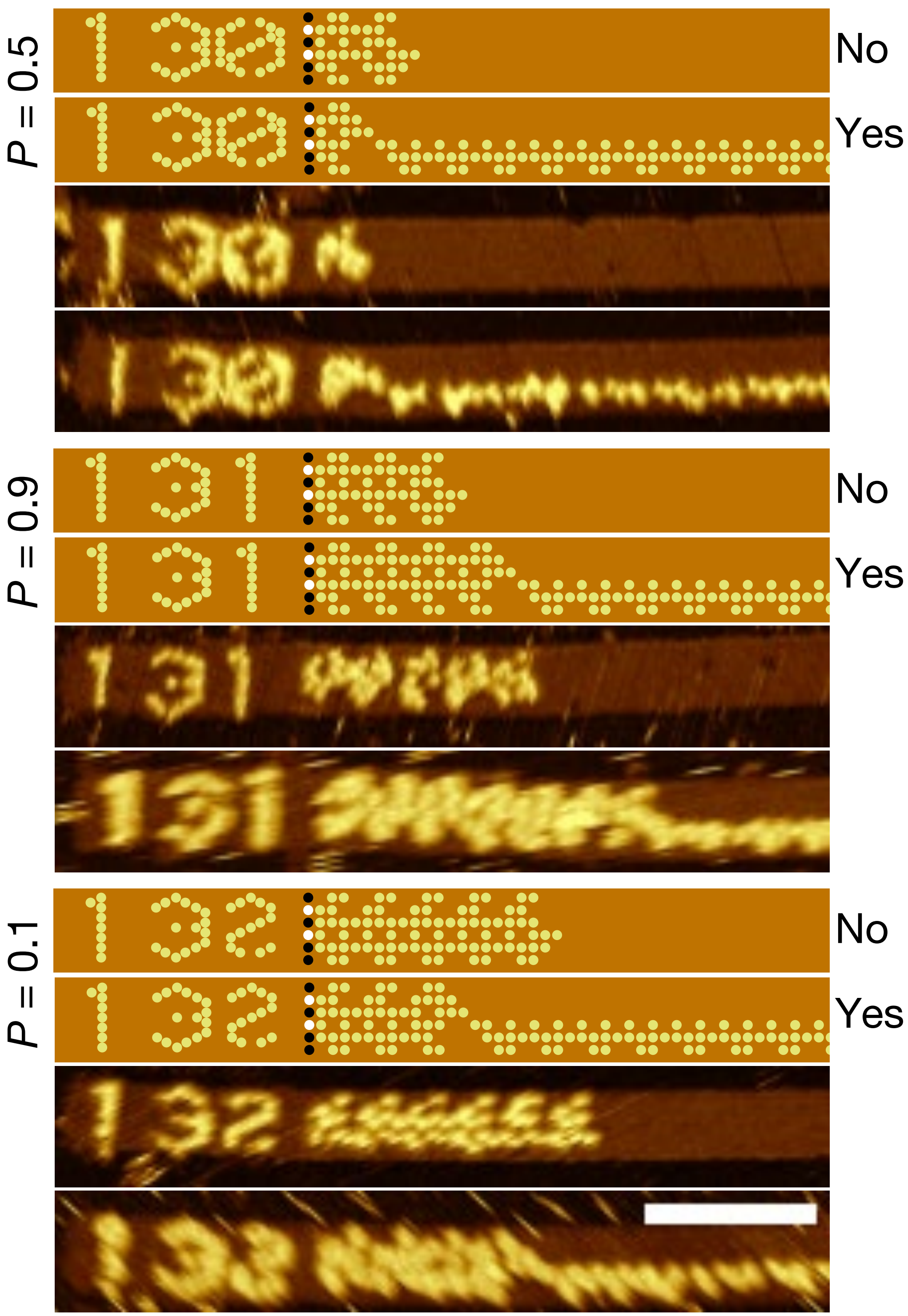


Tile-attachment error rate 0.03%±0.002
Number of tiles attached 354,355

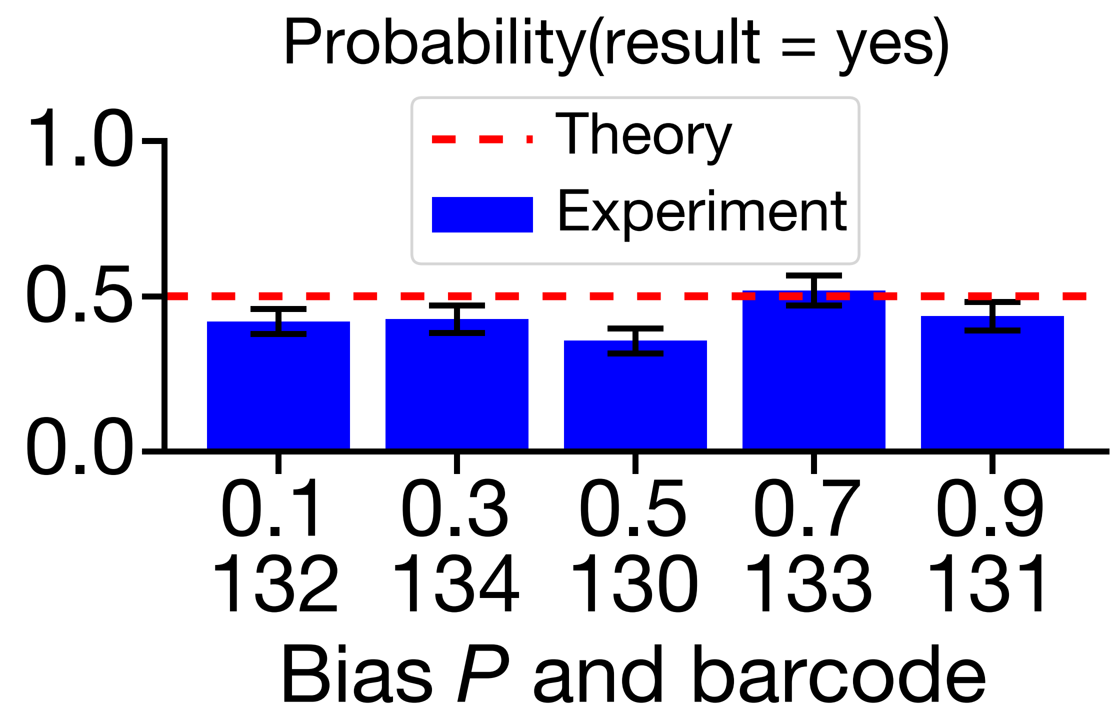
Unbiasing a biaised coin

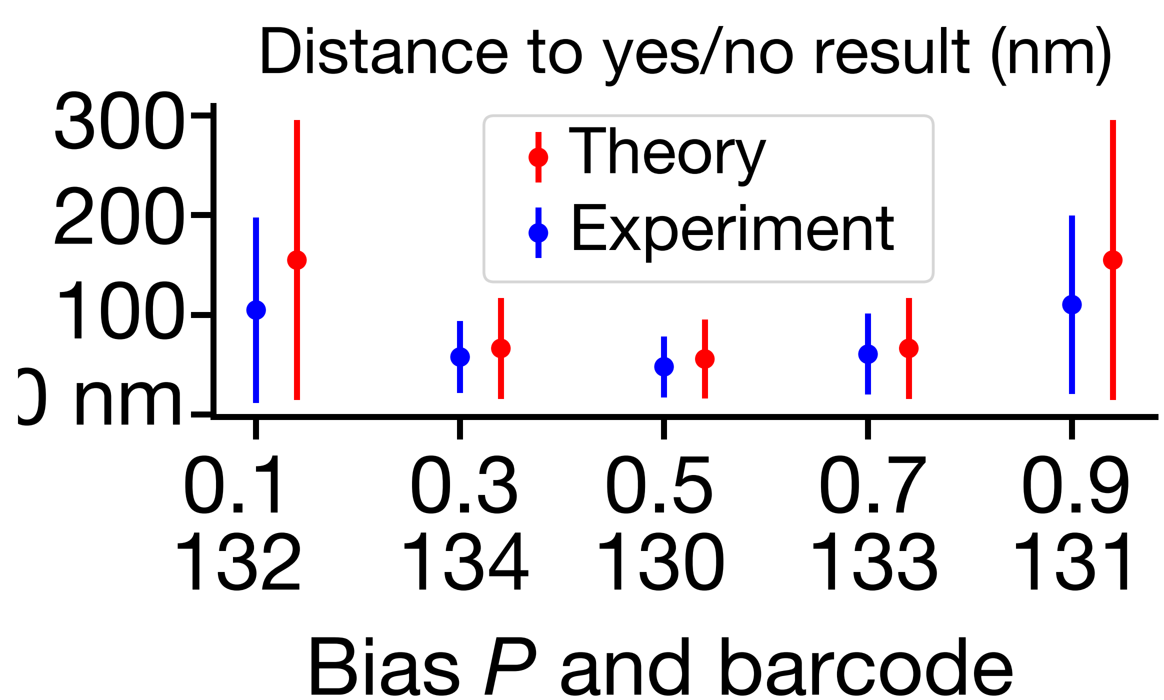


Unbiasing a biased coin



Tile-attachment error rate 0.01%±0.001
Number of tiles attached 545,785





Conclusion

- A 6-bits universal "efficient" DNA computer based on CA rule 110
- 3-5 years of hard work
- Beautiful results
- OPEN: interface computation for other circuits? reduce errors? have the circuits react to something?