Molecular computing with DNA: theory and implementation

Damien Woods
Joint work with: David Doty,
Cameron Myhrvold, Joy Hui, Felix Zhou,
Peng Yin, Erik Winfree

Maynooth University
National University of Ireland Maynooth

Hamilton Institute

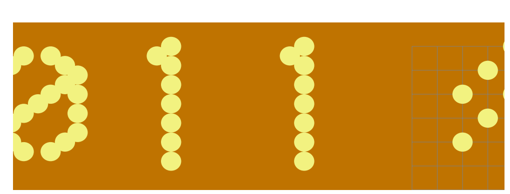
Caltech

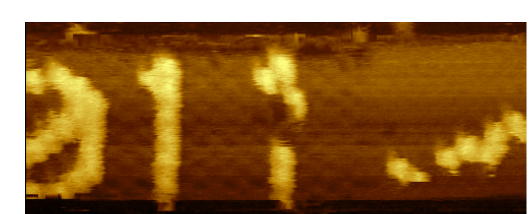
Inria

UC Davis

Harvard

Theorem: Let T be a tile assembly system …

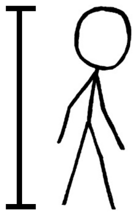


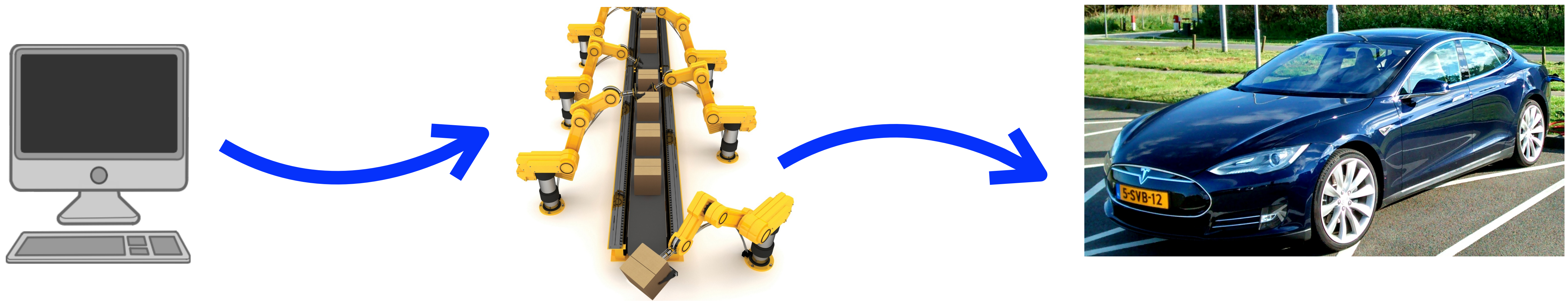


Building stuff

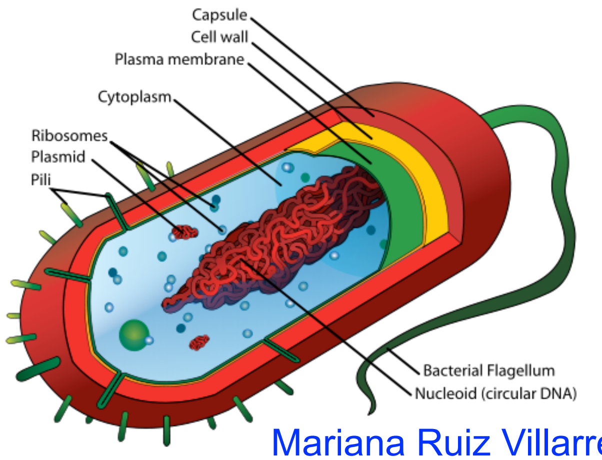
Ljubljana Marshes Wheel. 5k years old

Newgrange, Ireland. 5.2k years old

- **Building stuff by hand**: use tools! Great for scale of $10^{+/-2}$ x 
- **Building tools that build stuff**: specify target object with a computer program that then controls the manufacturing process

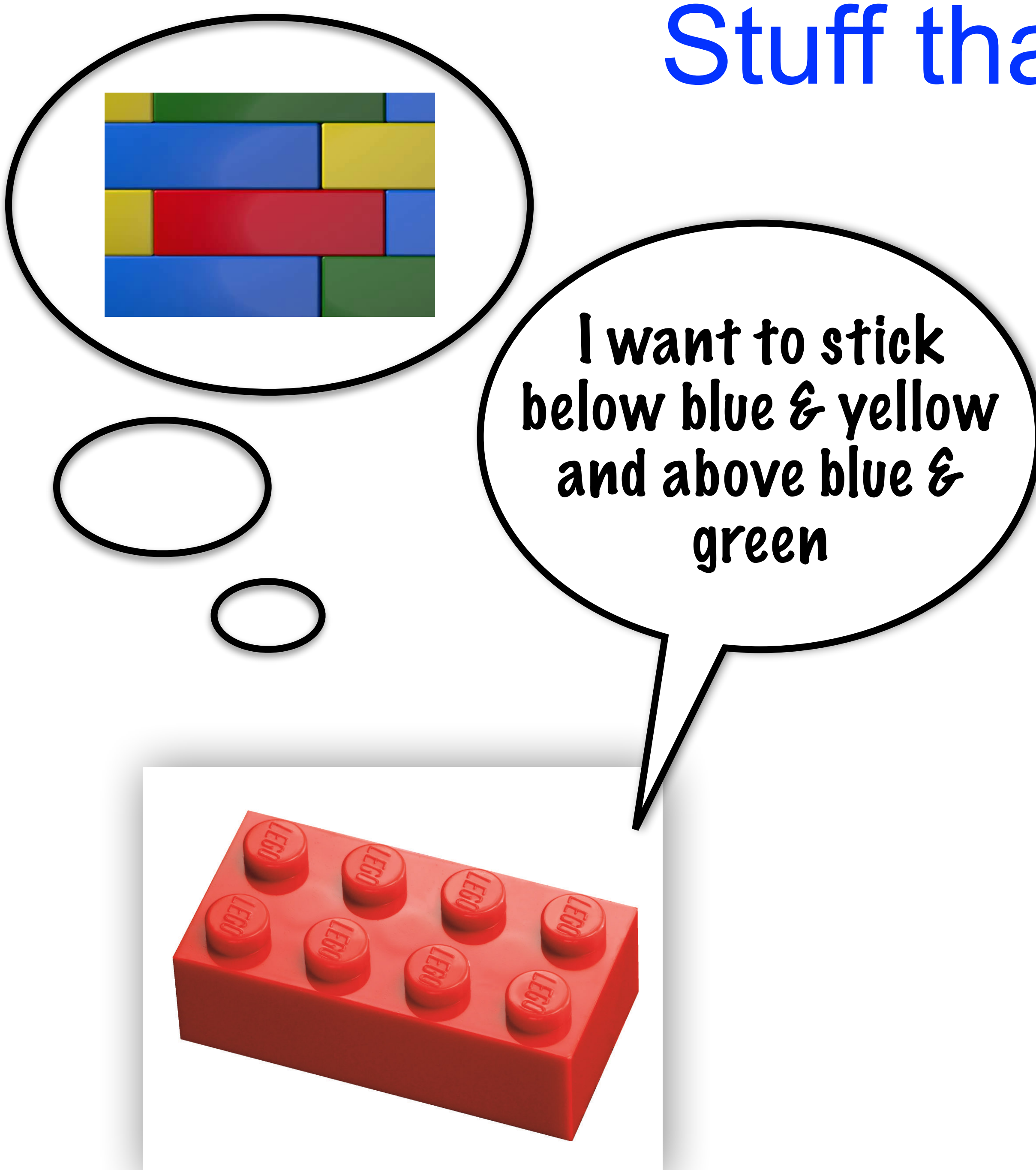


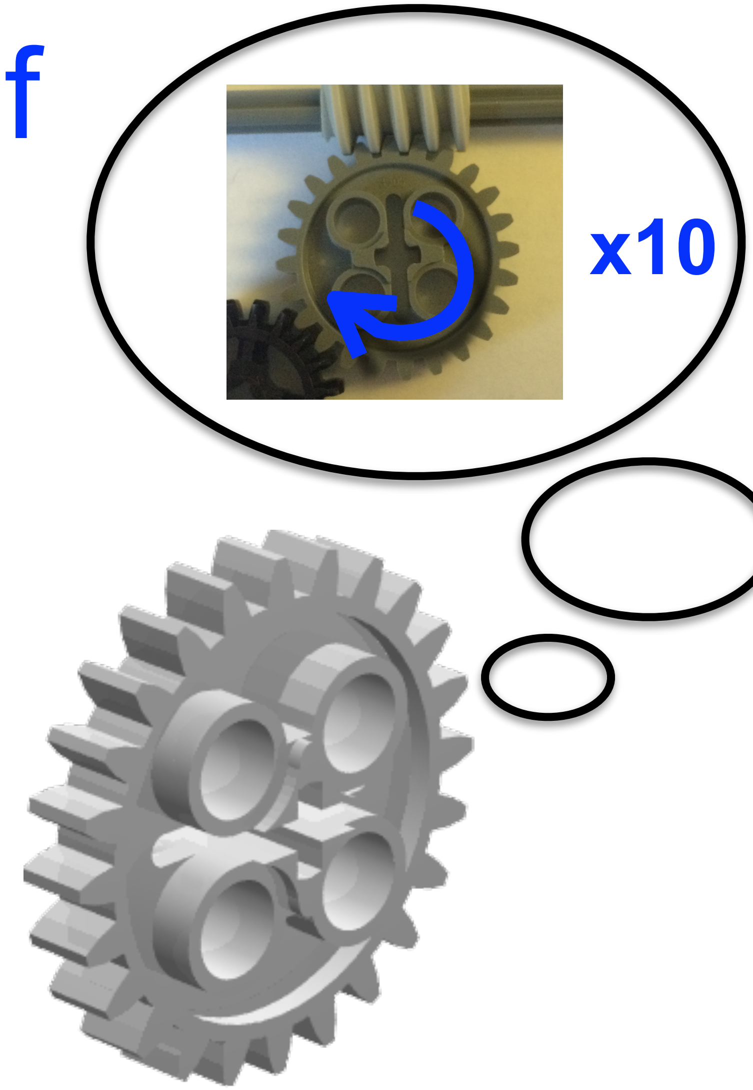
- Programming **stuff to build itself**: for building stuff in small wet places where our hands or tools can't reach



Mariana Ruiz Villarreal

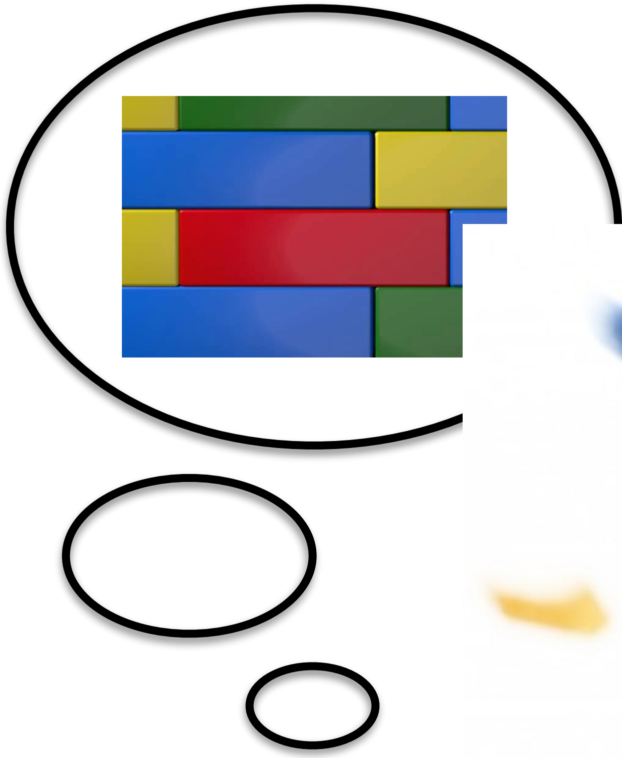
Stuff that builds itself





- Today you'll hear about self-assembling molecules that compute as they build themselves

Stuff that builds itself



- Today you’ll hear about self-assembling molecules that compute as they build themselves

Stuff that builds itself

- Today you'll hear about self-assembling molecules that compute as they build themselves

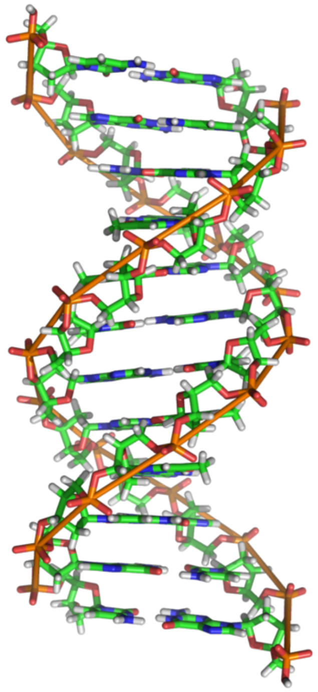
Biology is an expert at nanoscale engineering

- Evolution: a long-term random and unpredictable process used by Biology, not a good way engineer (Too slow. What are the principles? Not reproducible. Caveat: Directed evolution in a controlled environment)
- We'd like to be able to design from the ground up:
  - using materials and processes we understand
  - using a hierarchy to handle complexity
  - systems that have the potential for computation

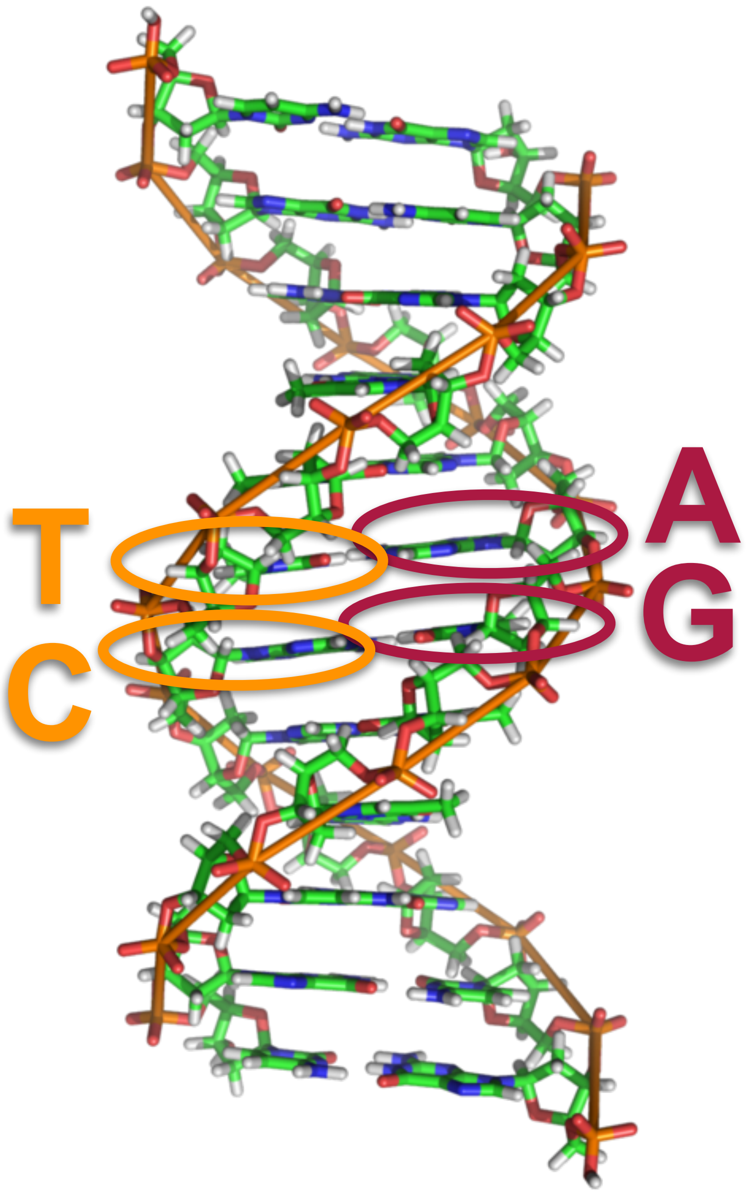
Newly hatched zebrafish,
3 days after zygote

Material

- DNA is a material that fits the bill! (Thanks Biology!)
- DNA is so predictable and well-understood that we can use it like a kind of nanoscale Lego

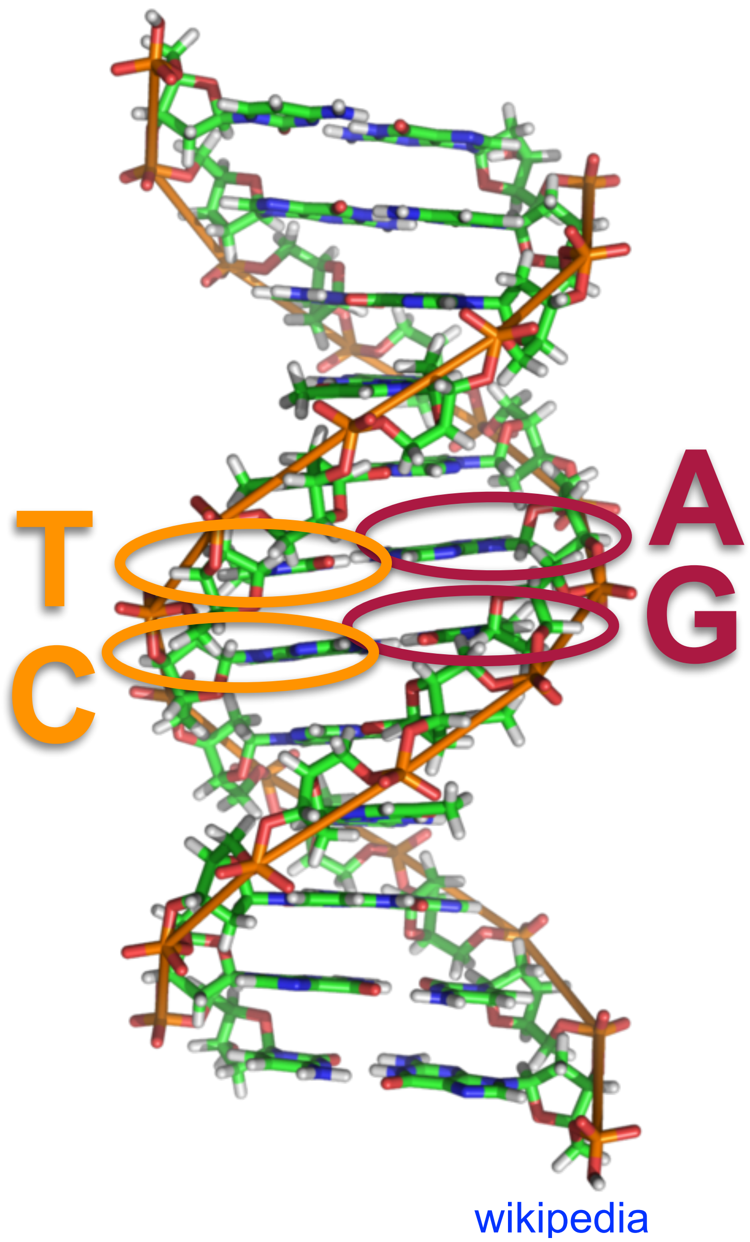


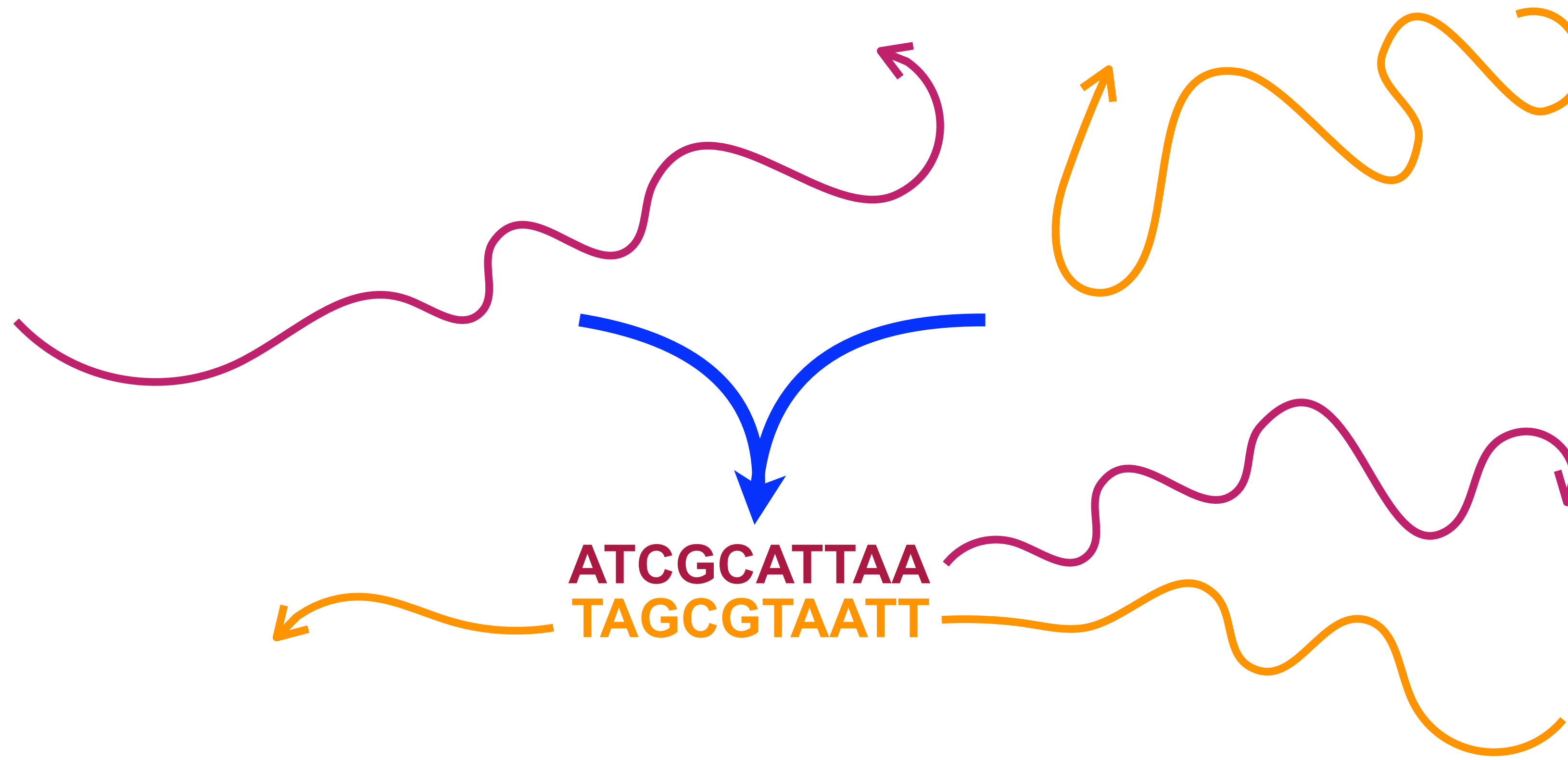
DNA & DNA tiles



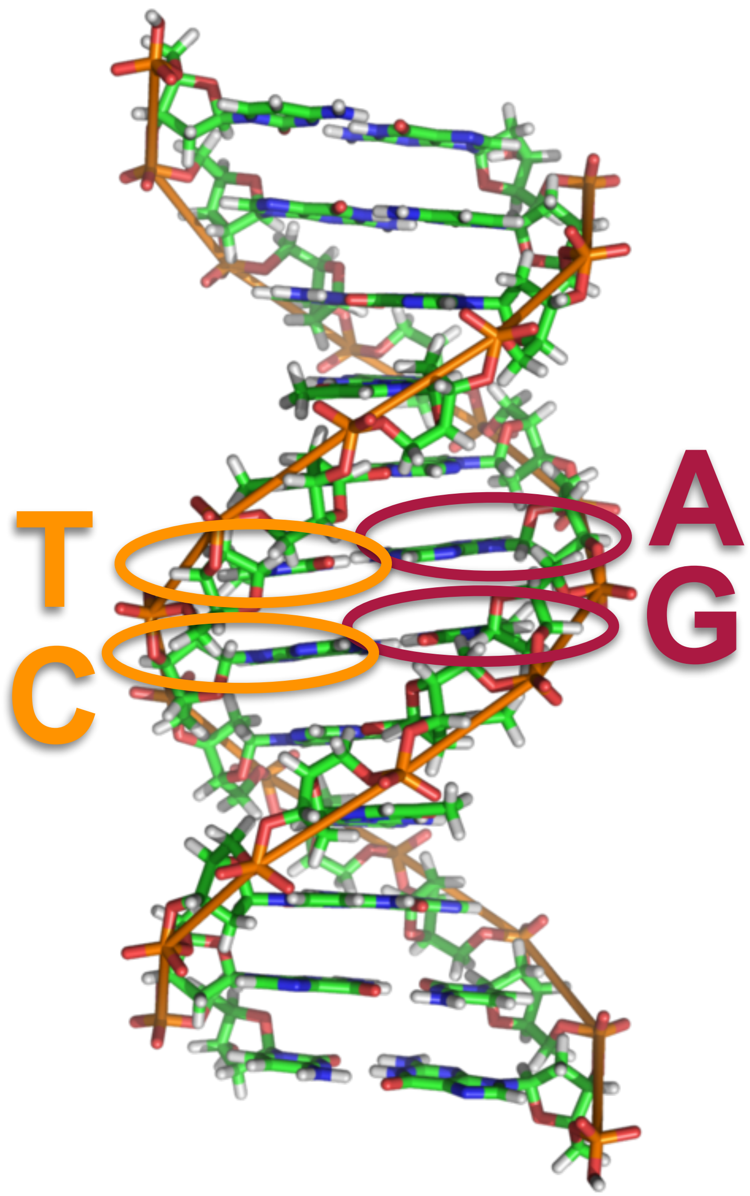
wikipedia

DNA & DNA tiles

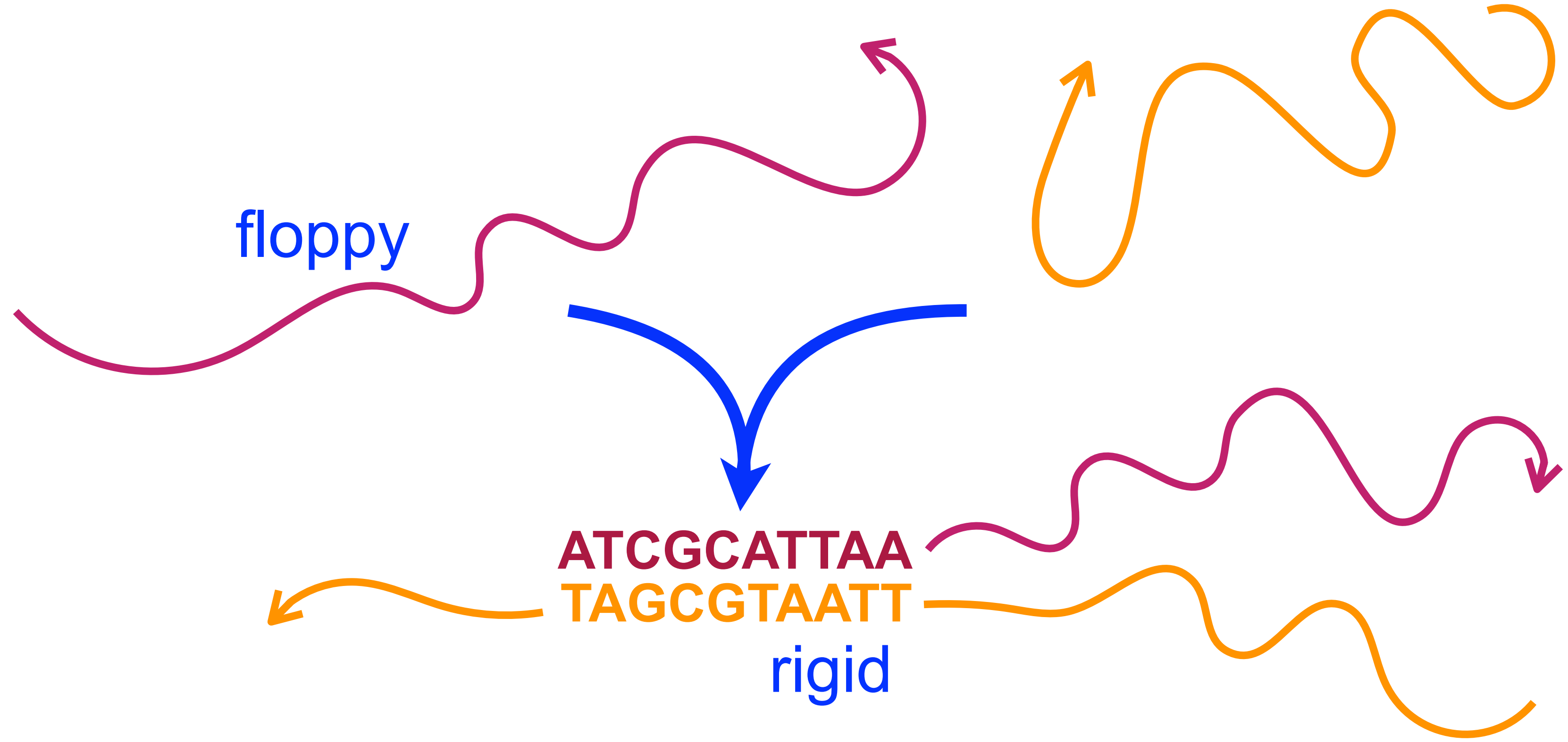




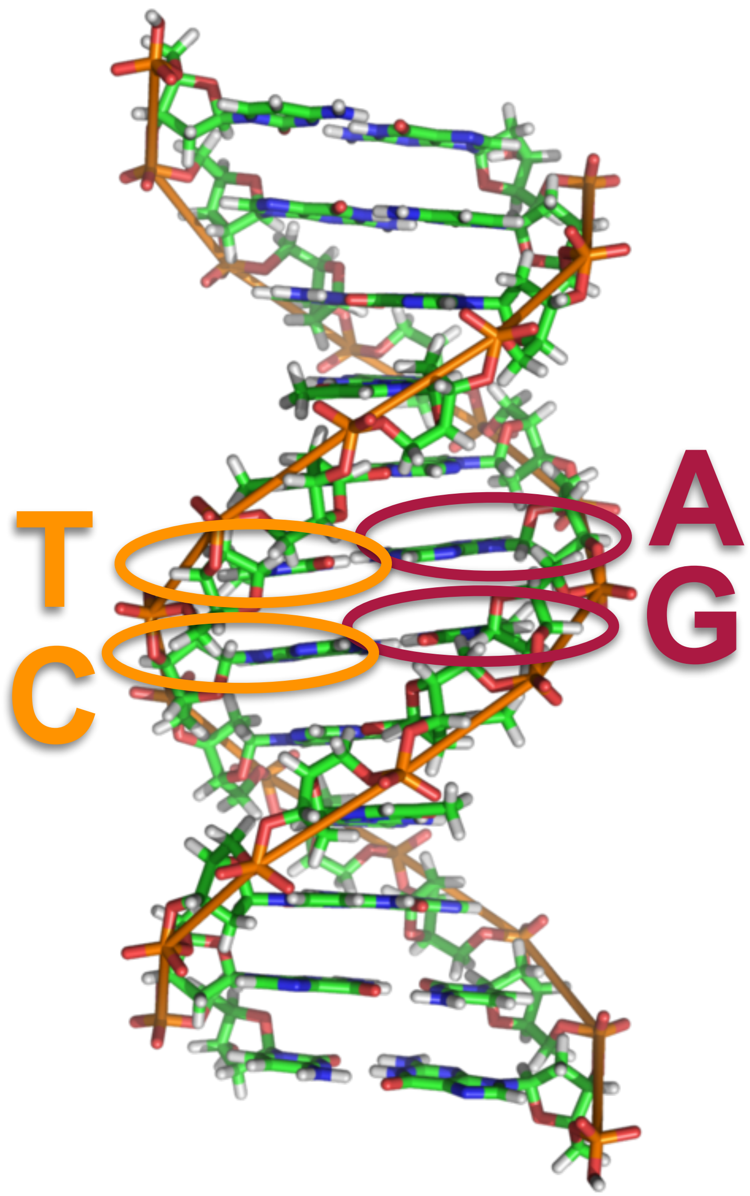
DNA & DNA tiles



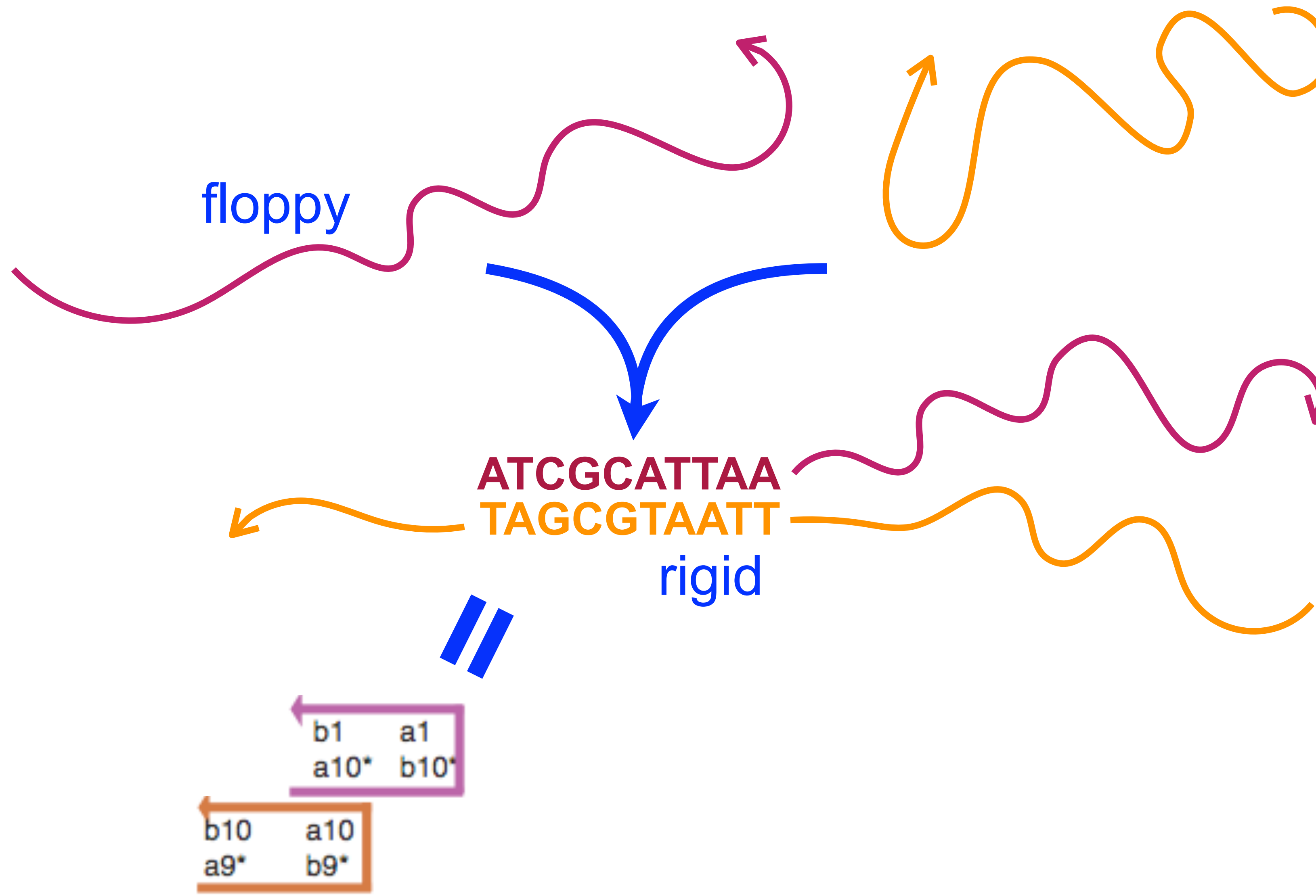
wikipedia



DNA & DNA tiles

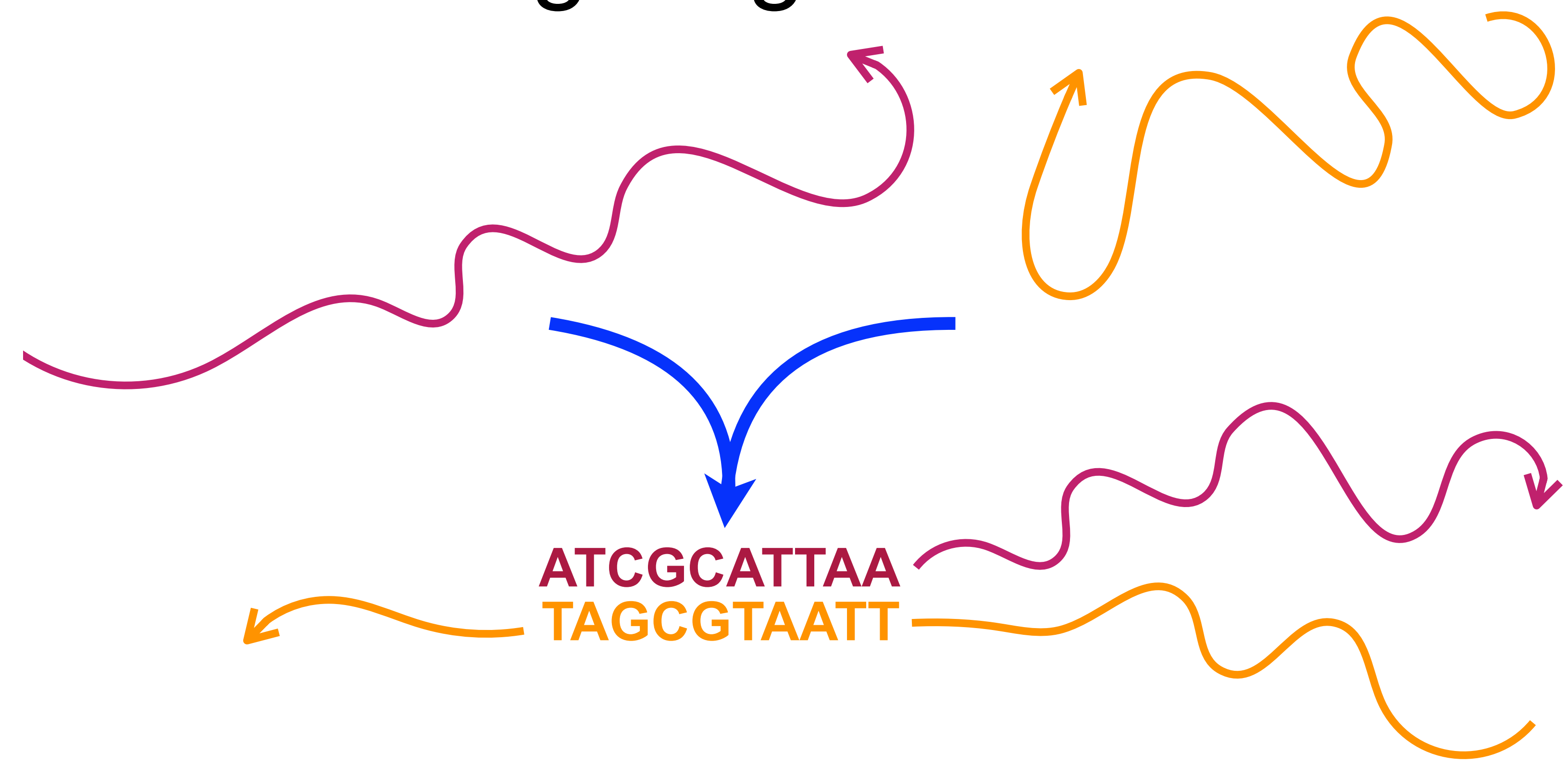


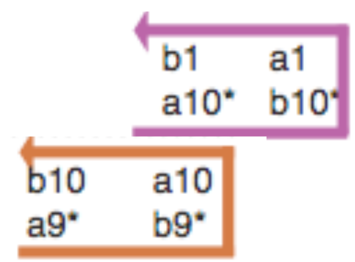
wikipedia

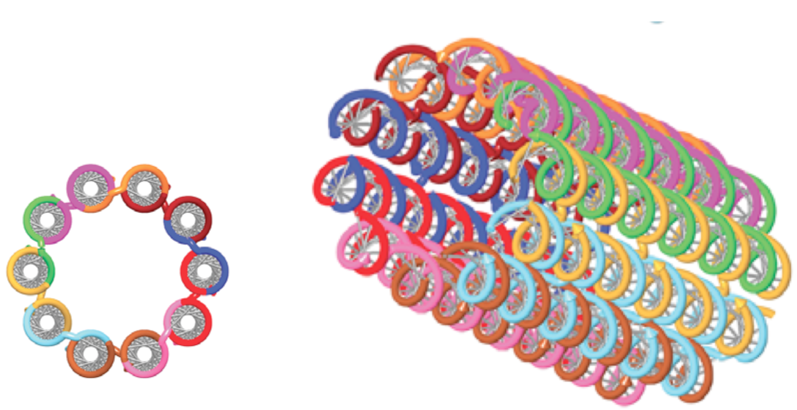


Example DNA nanostructure using single-stranded tiles

- Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008

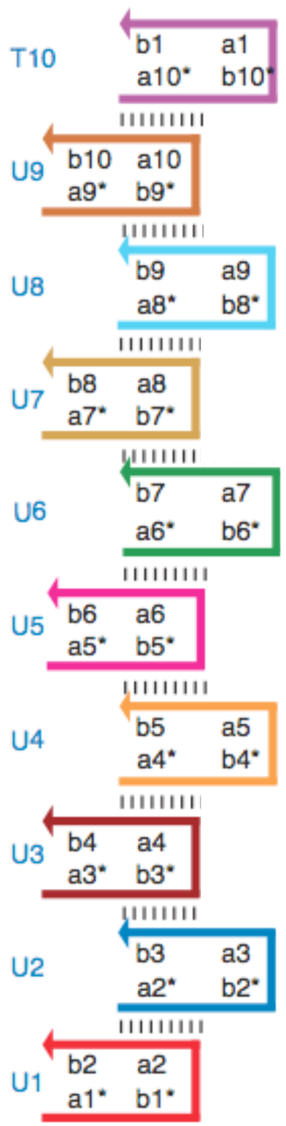


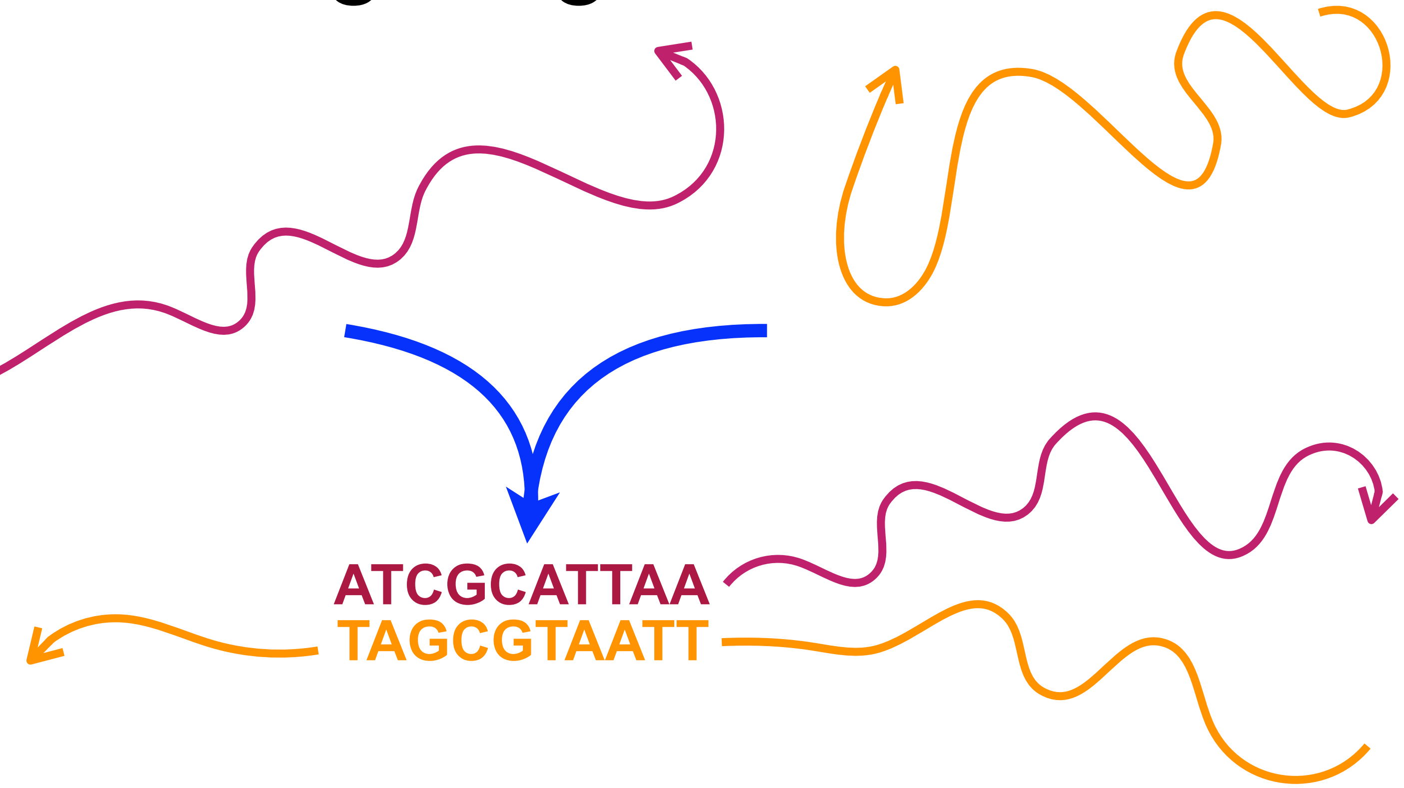


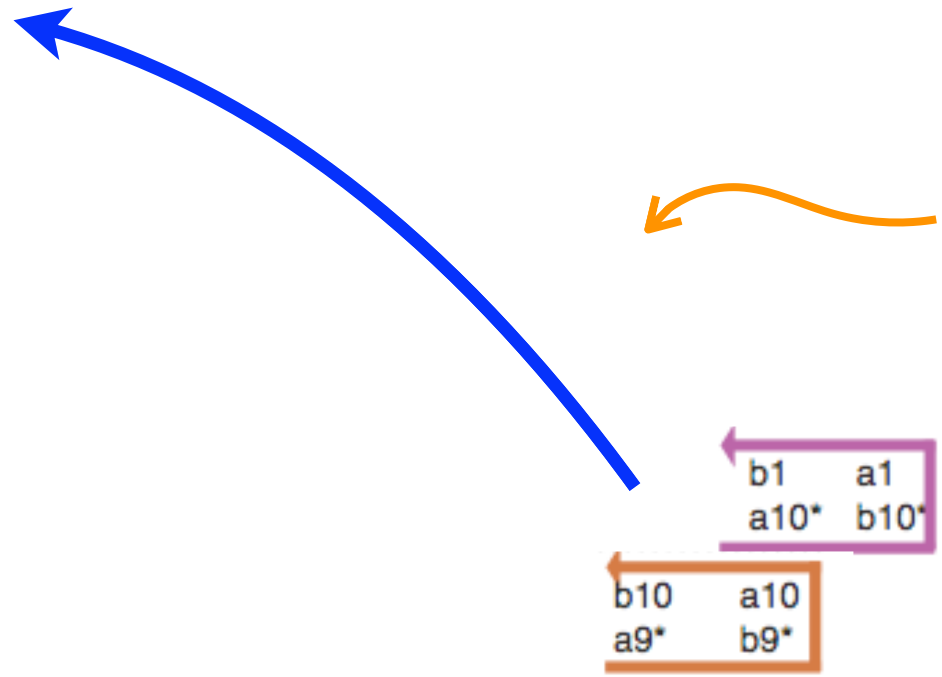


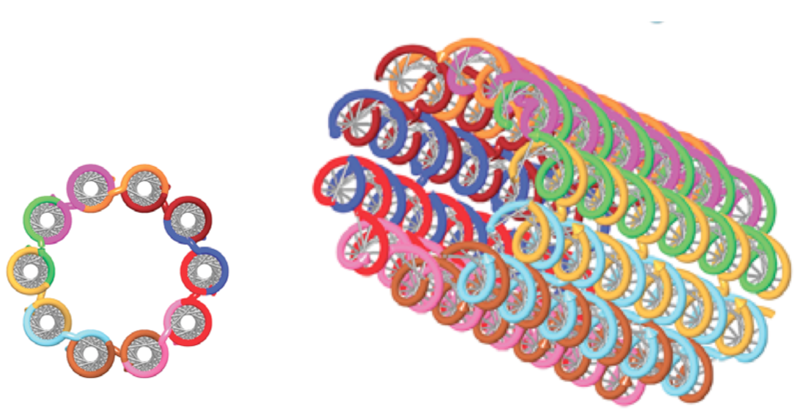
Example DNA nanostructure using single-stranded tiles

- Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008



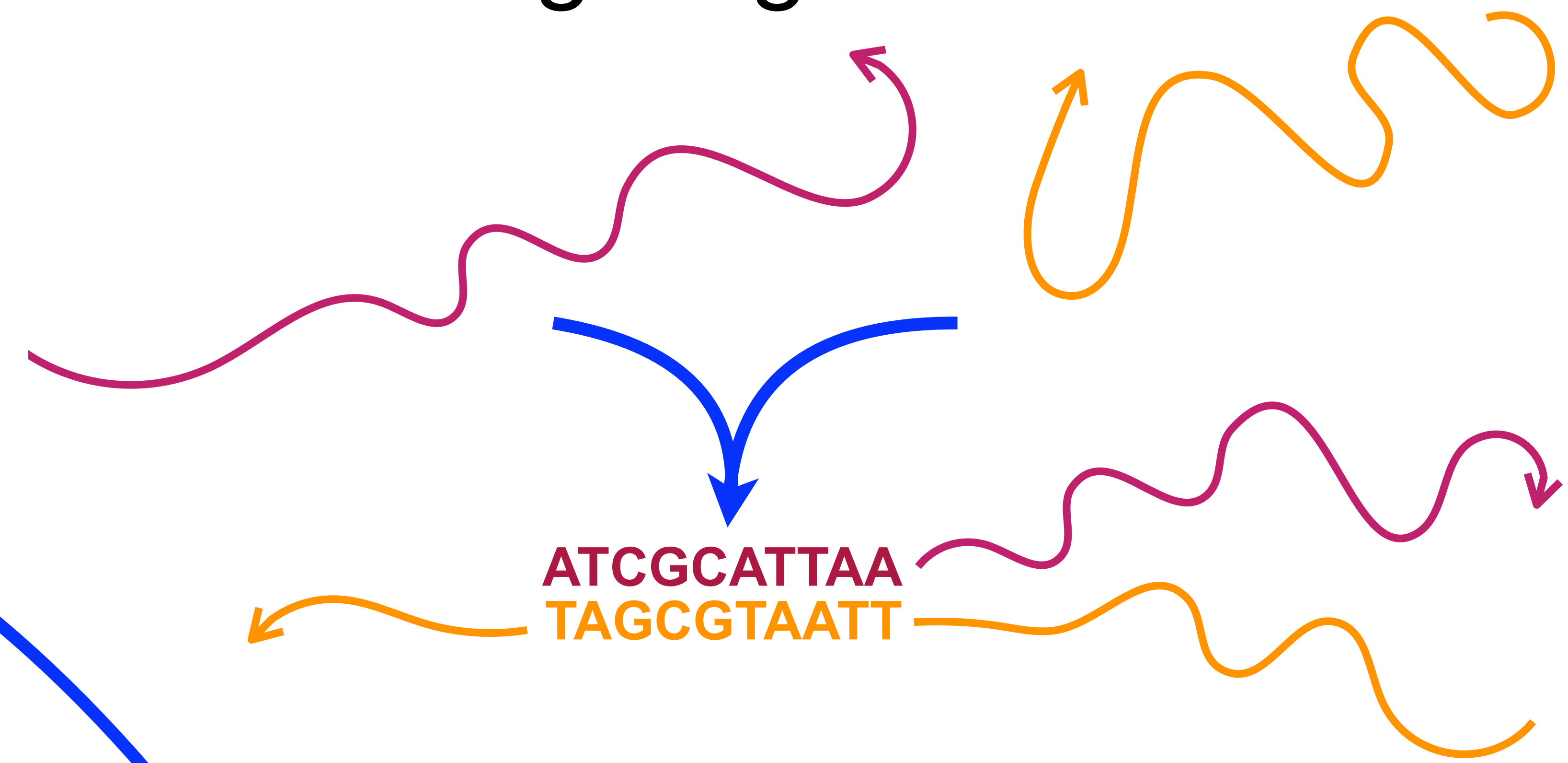


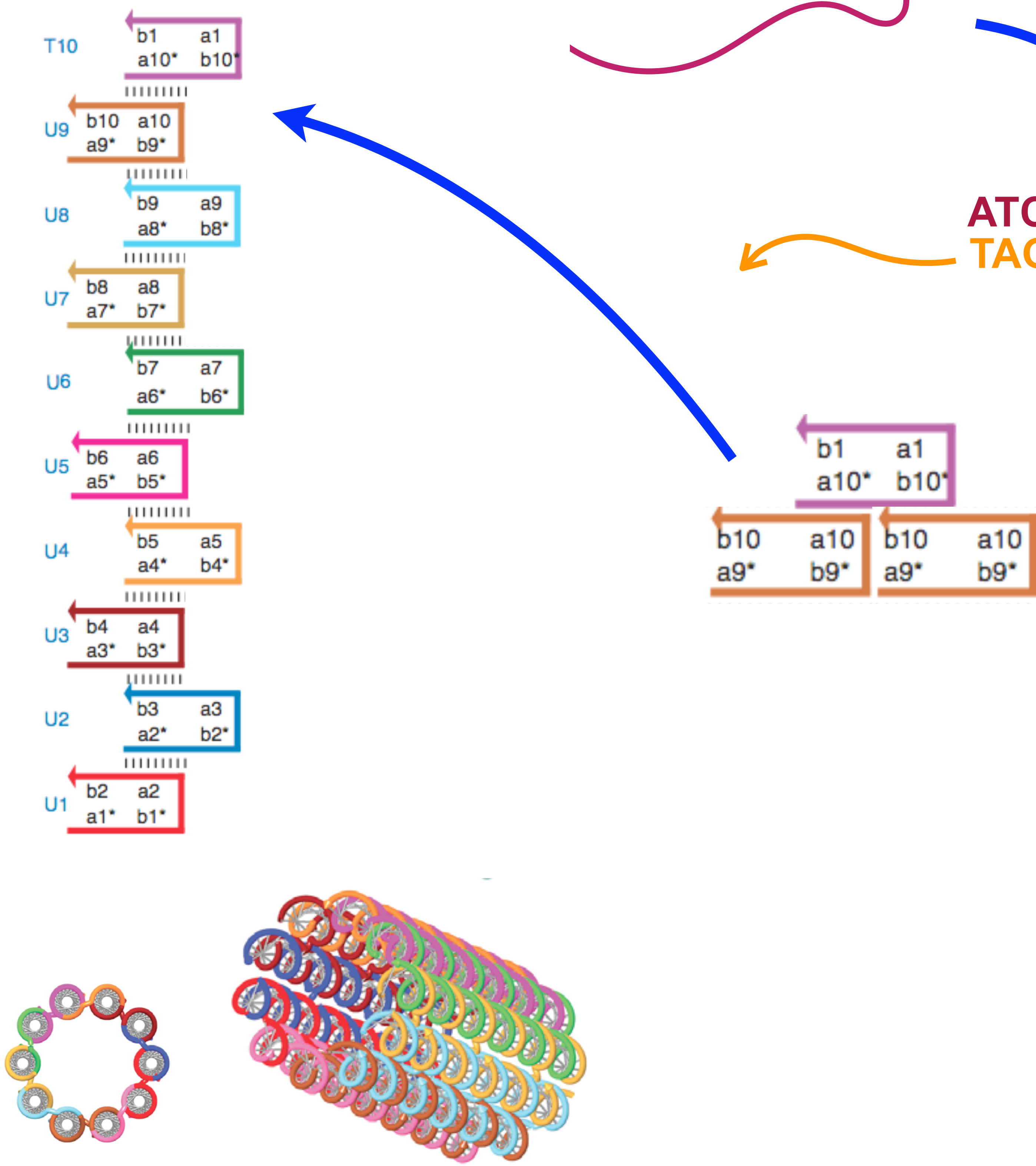




Example DNA nanostructure using single-stranded tiles

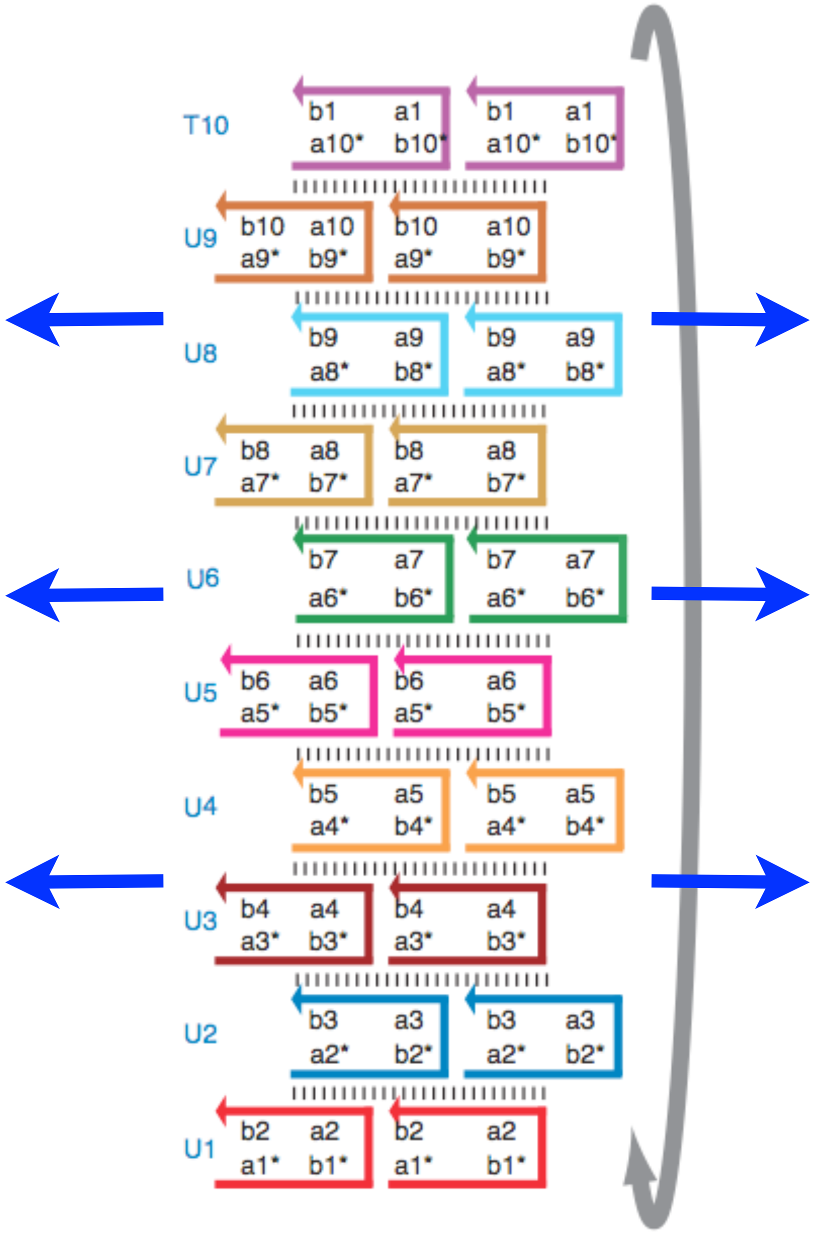
- Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008

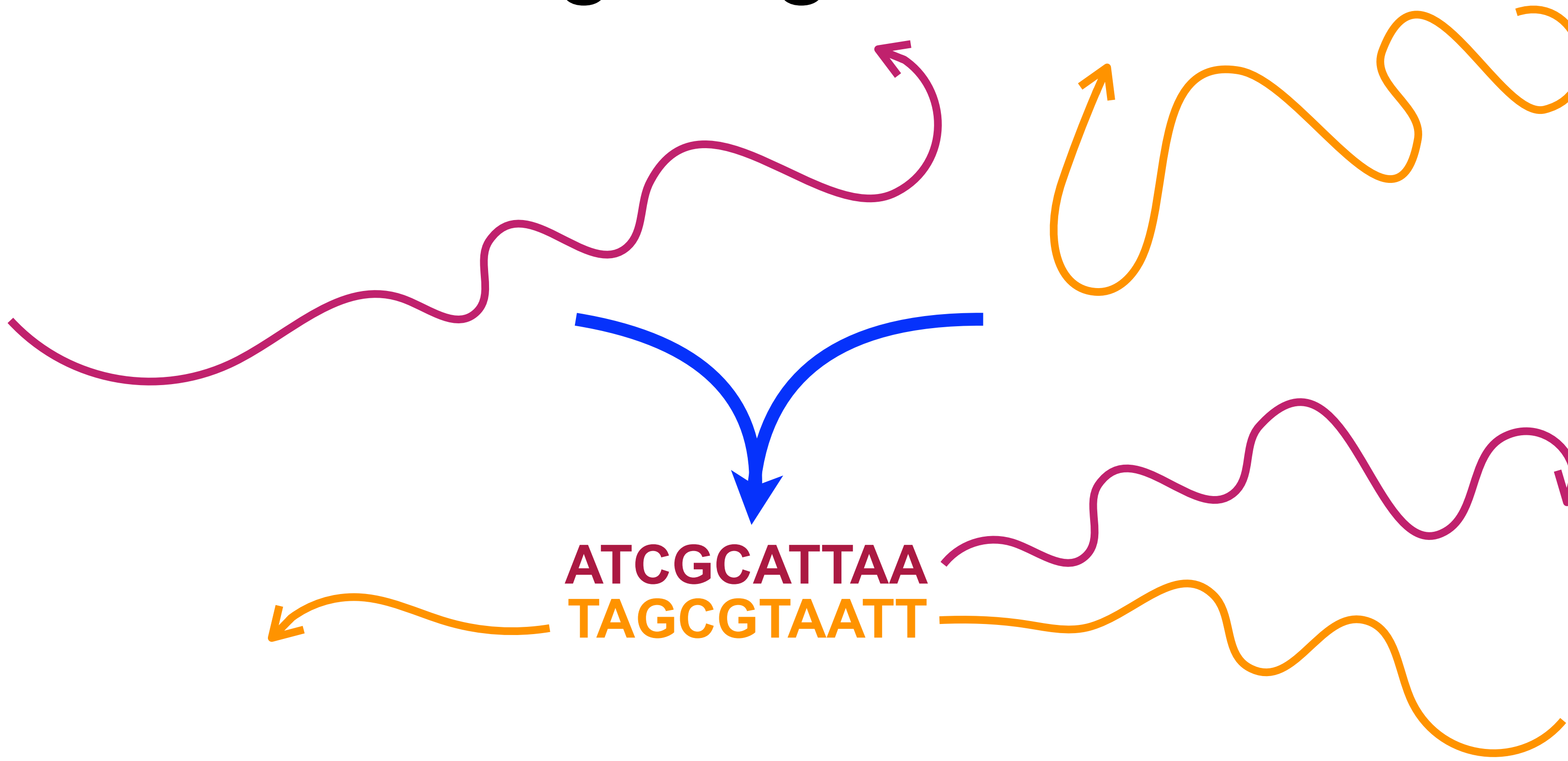


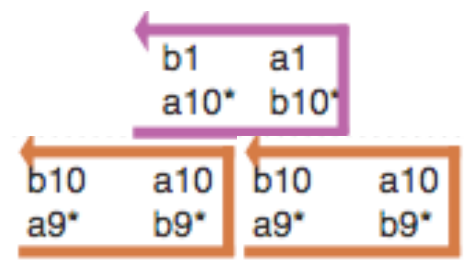


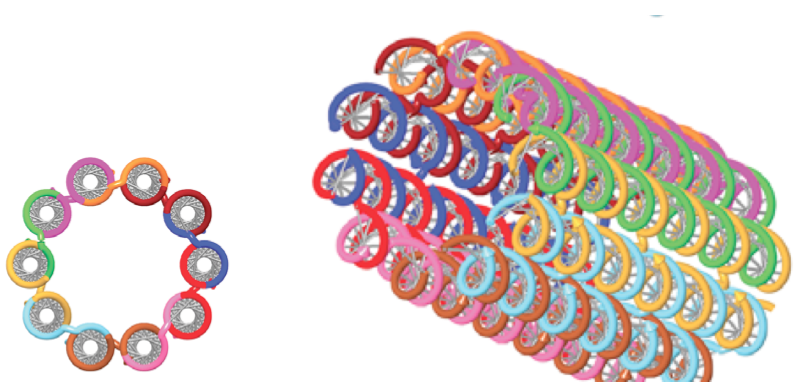
Example DNA nanostructure using single-stranded tiles

- Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008





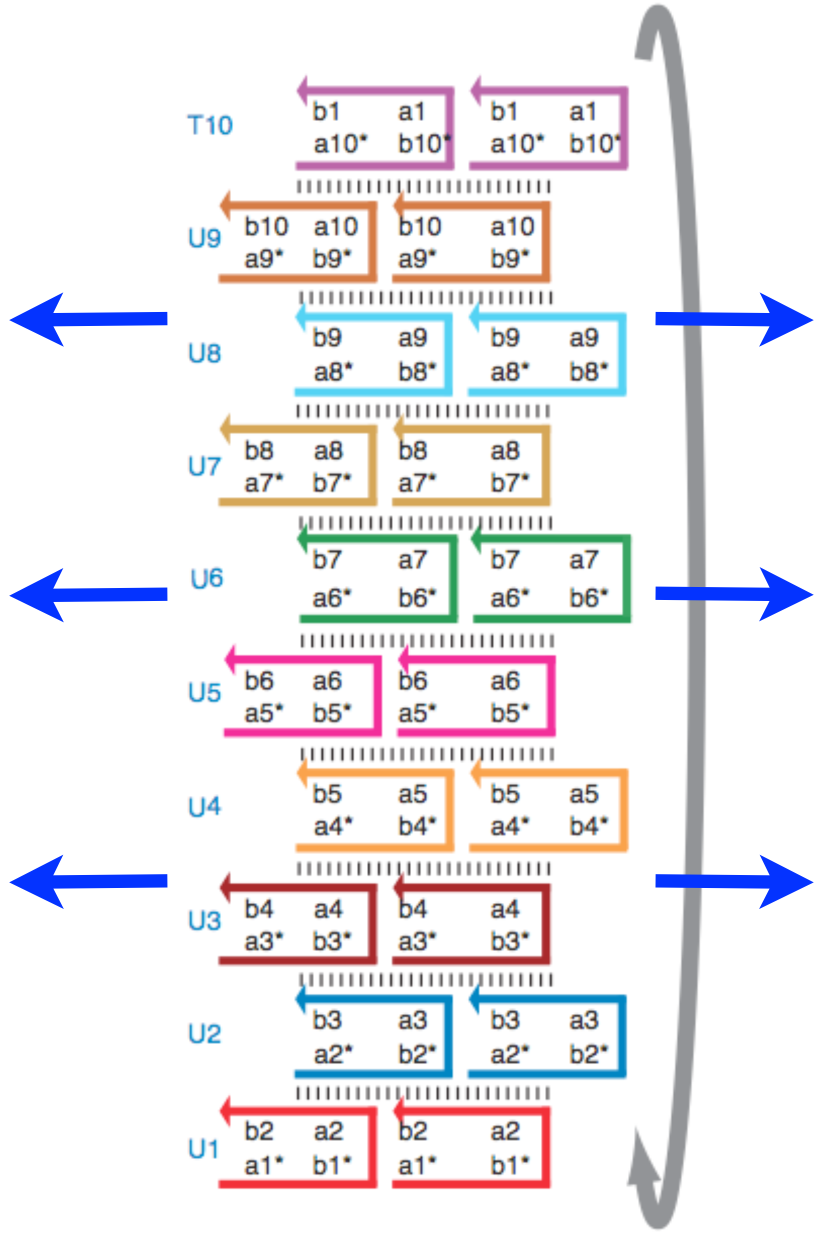


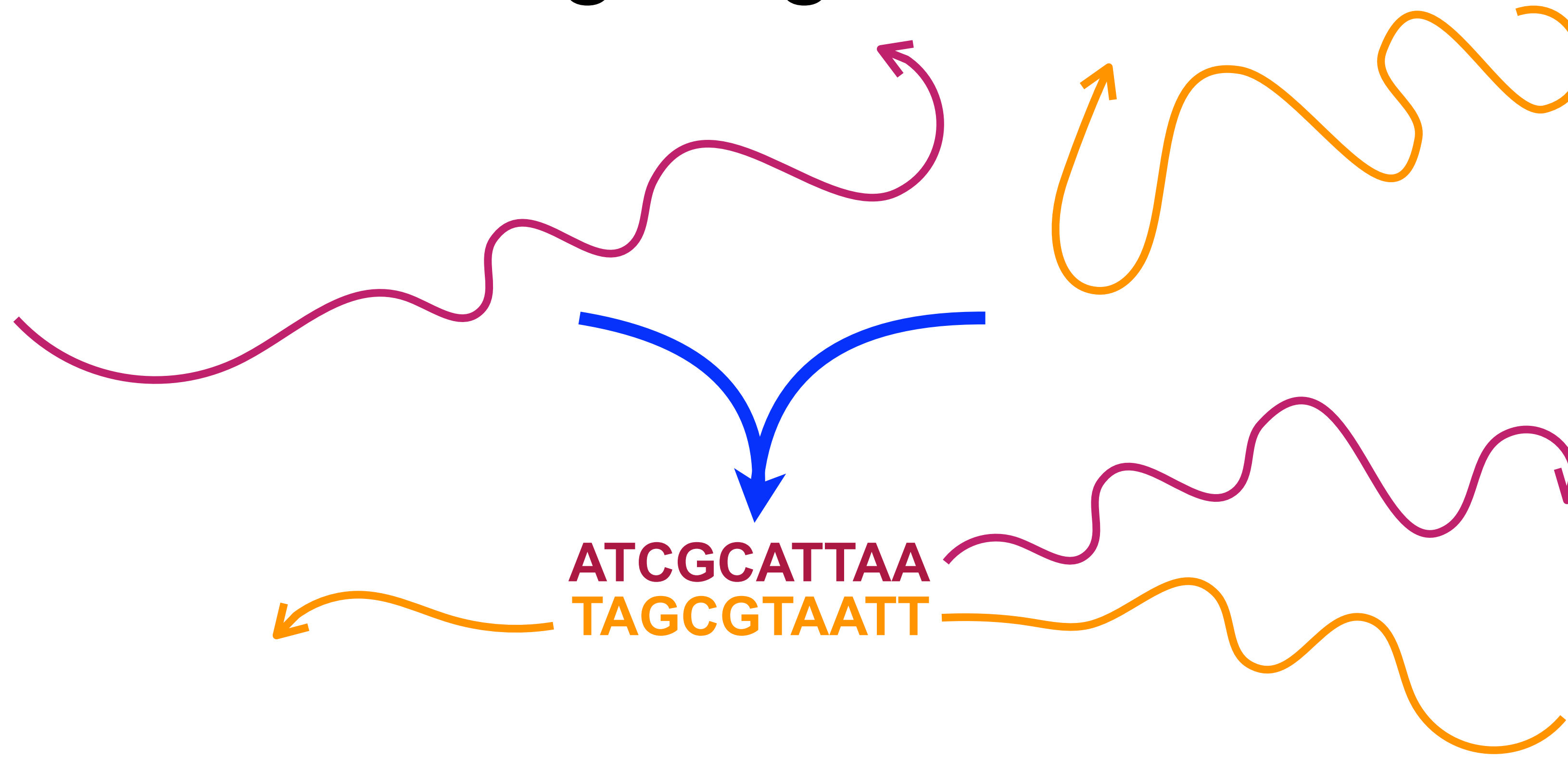


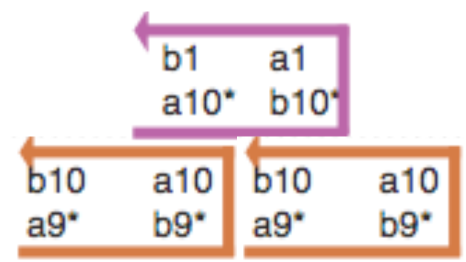
DNA nanotube!

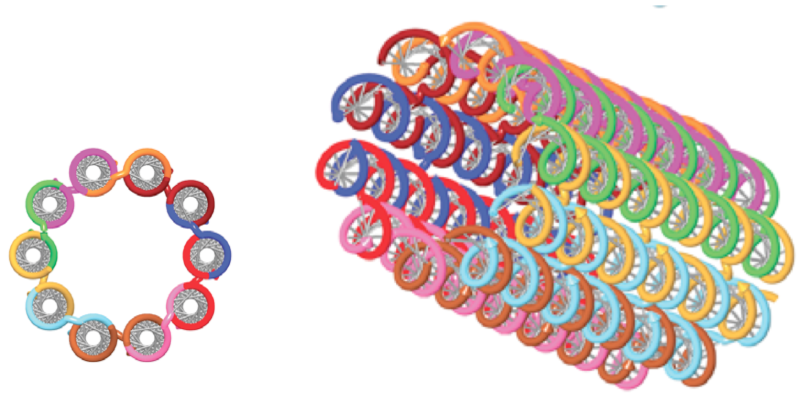
Example DNA nanostructure using single-stranded tiles

- Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008

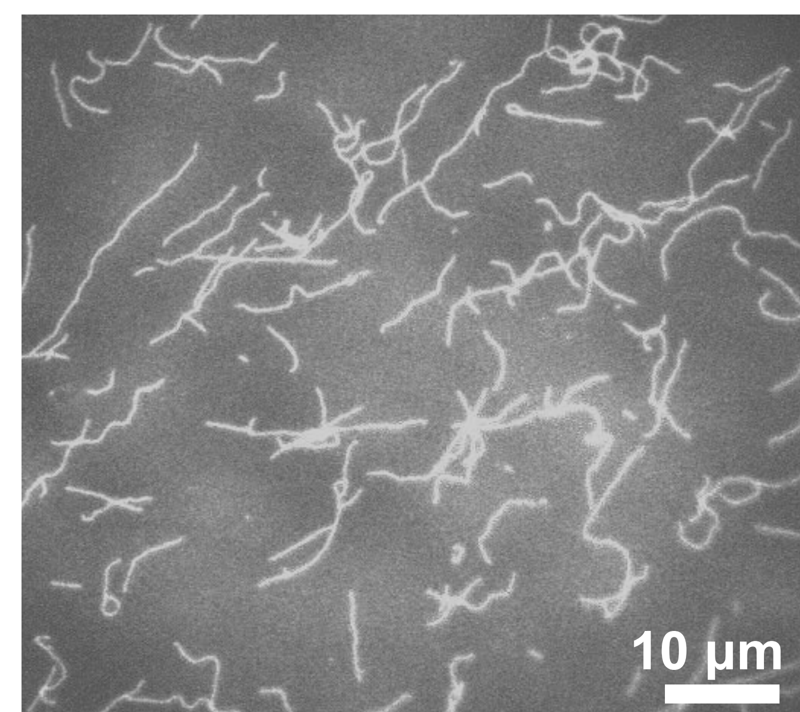








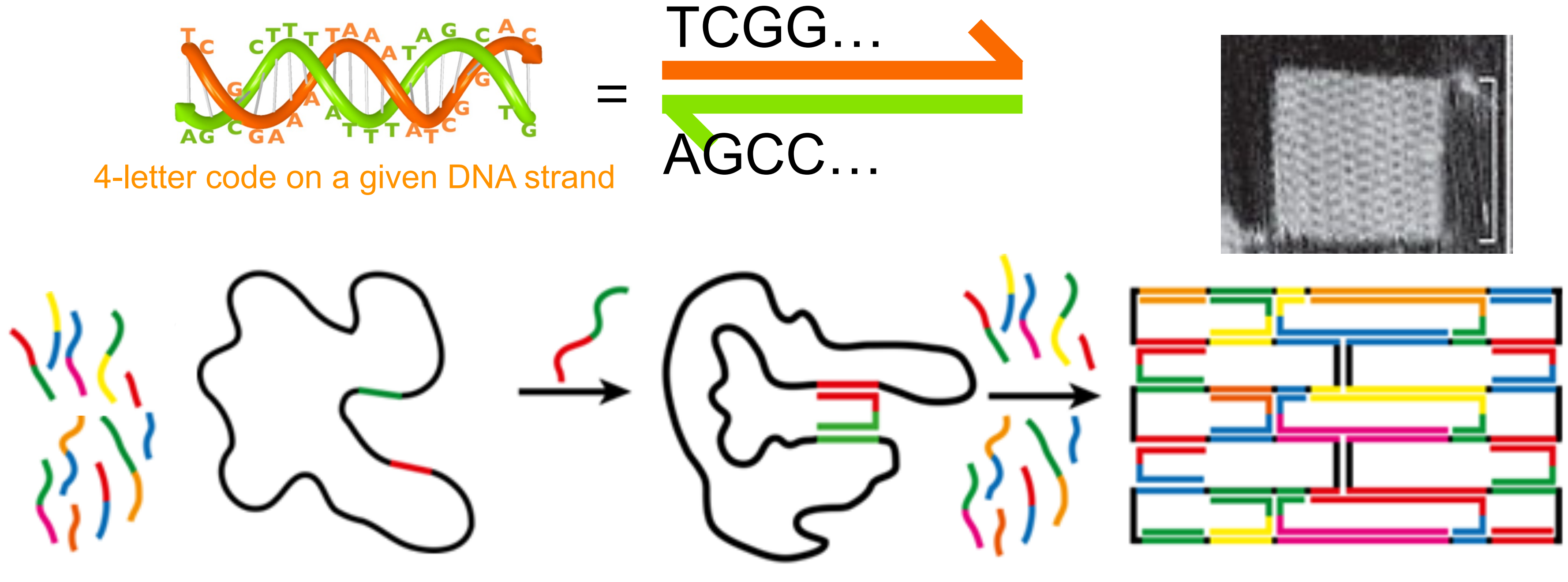
DNA nanotube!



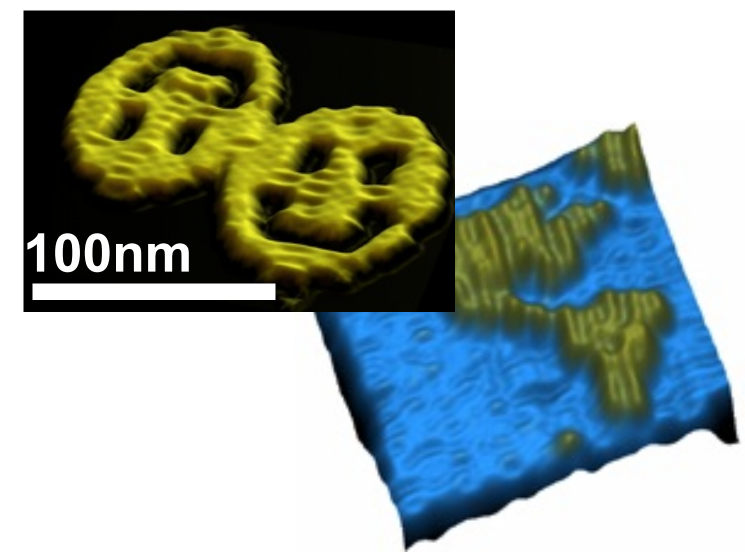
D Woods. Flourscently labelled nanotubes

We made a “DNA nanotube” using small DNA strands, what else can strands do?

Background: DNA nanostructures



DNA origami

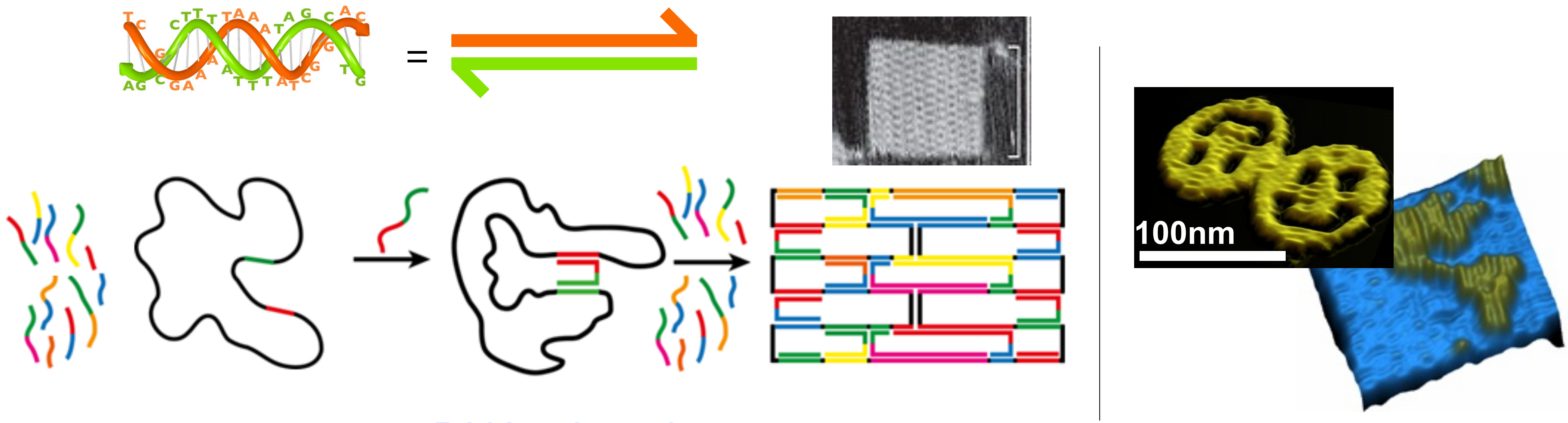


Rothemund. 2006 Nature

Example DNA nanostructure: DNA origami

Movie by Shawn Douglas

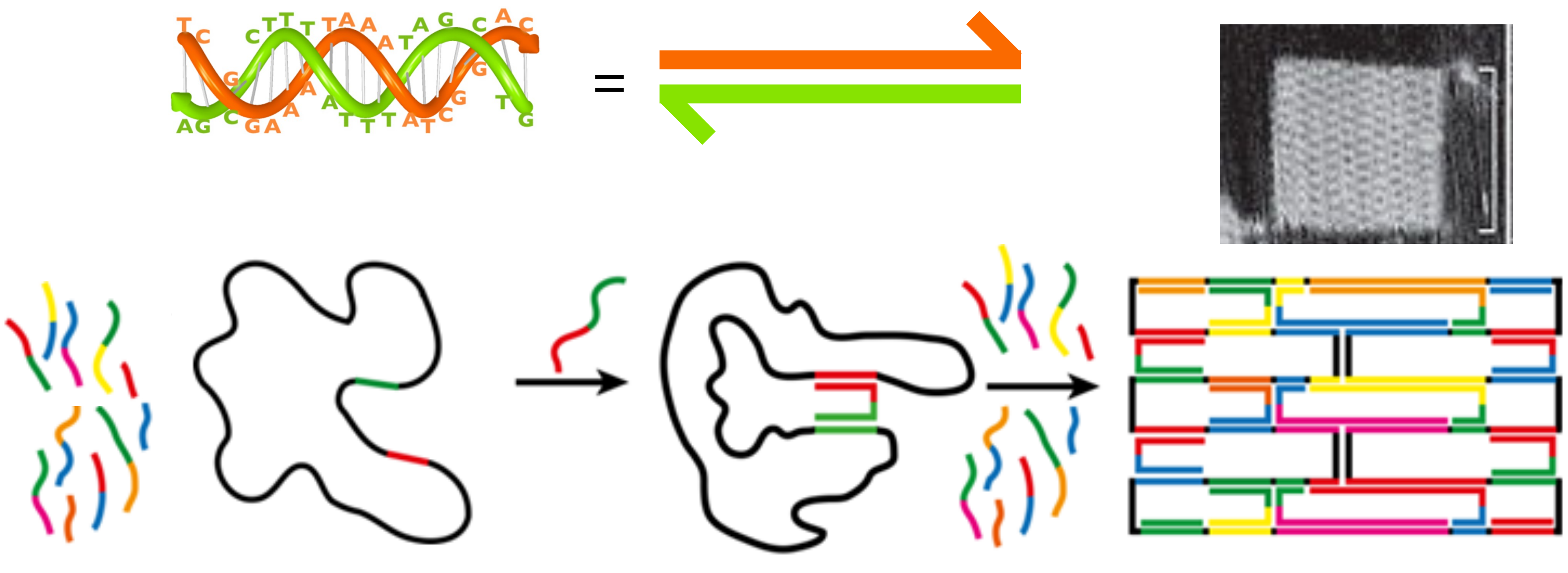
Background: DNA nanostructures



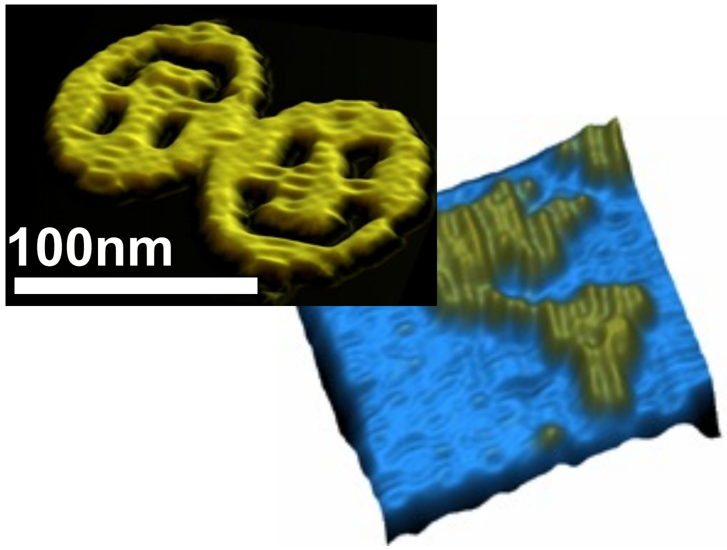
DNA origami

Rothemund. 2006 Nature

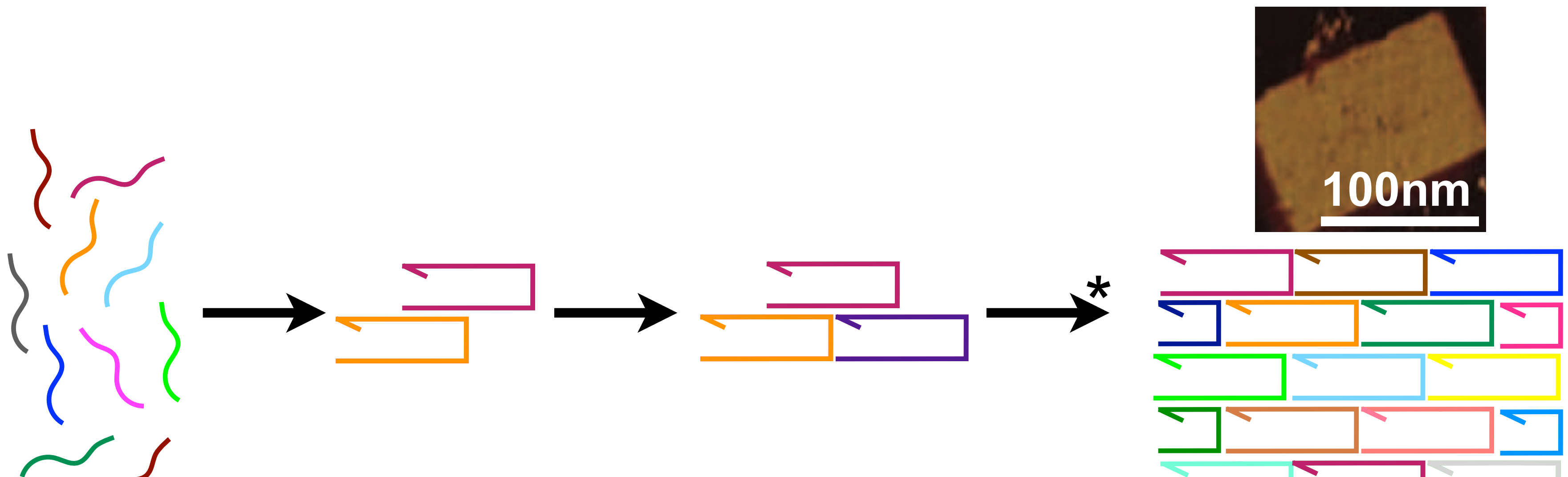
Background: DNA nanostructures



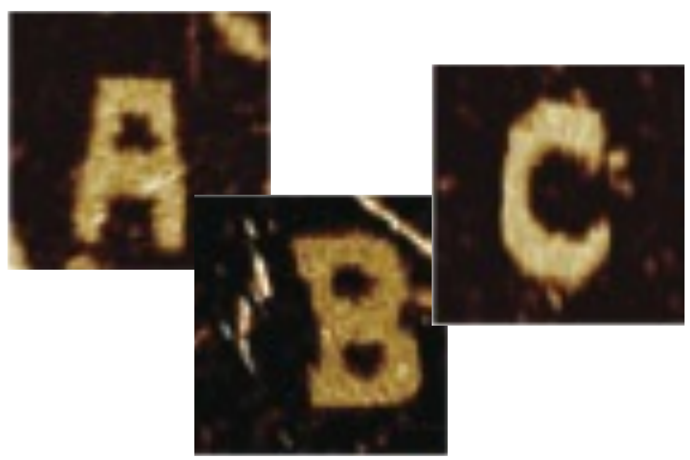
DNA origami



Rothemund. 2006 Nature

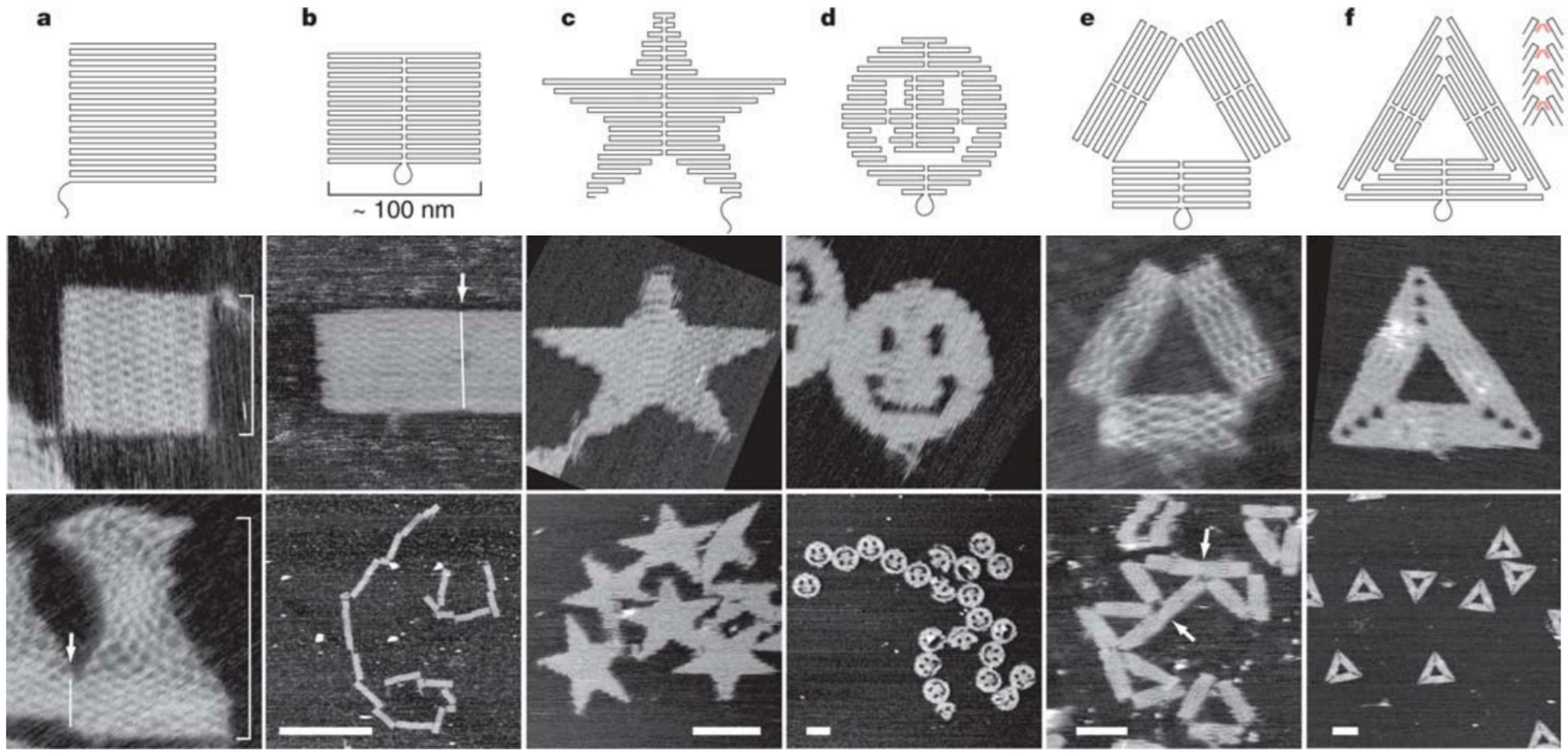


DNA single-stranded tiles



Wei, Dai, Yin. 2013 Nature

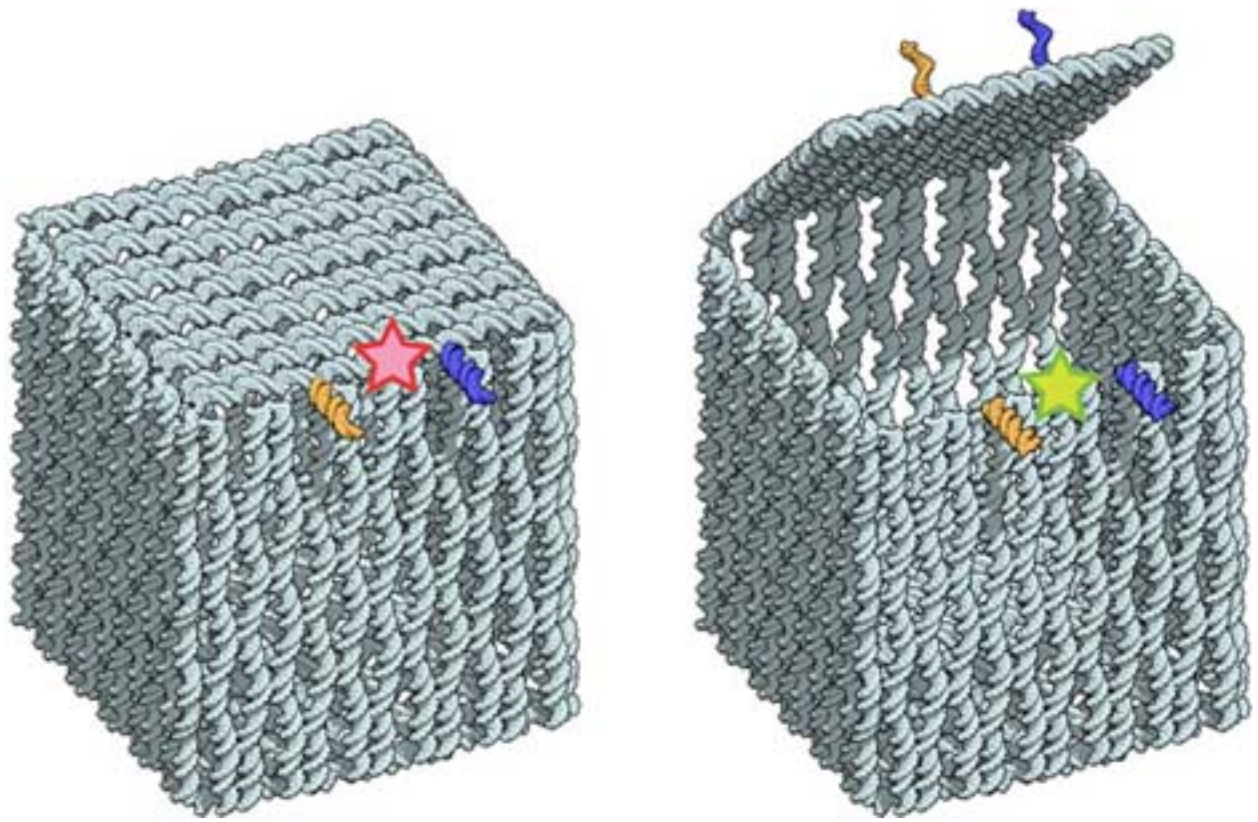
DNA nanostructure examples



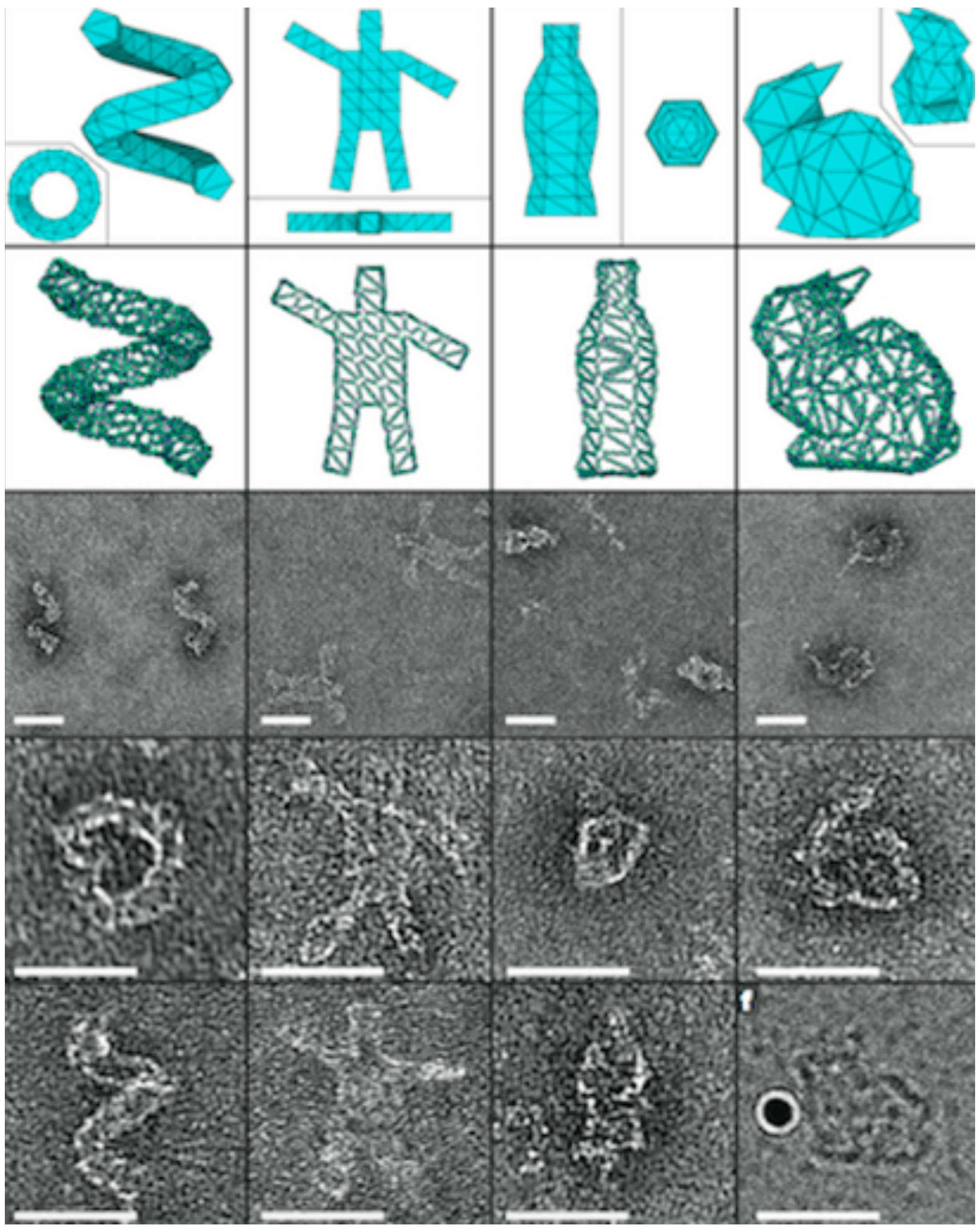
DNA origami: Rothemund, Nature 2006

DNA single-stranded tiles: Wei, Dai, Yin, Nature 2012

3D shapes made out of DNA



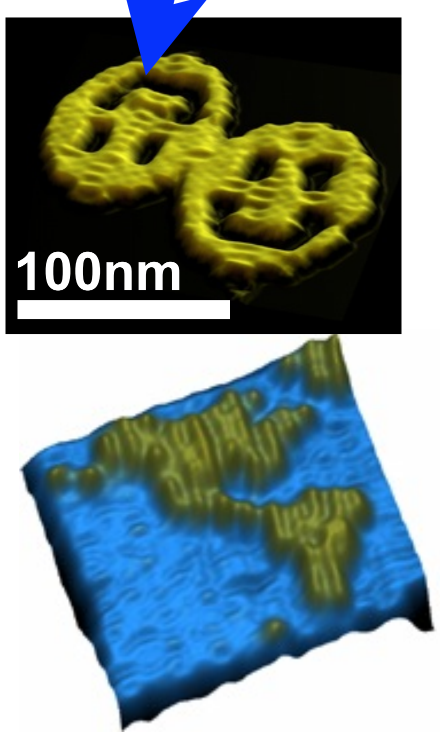
DNA origami box (that opens)
Andersen et al. Nature 2009.



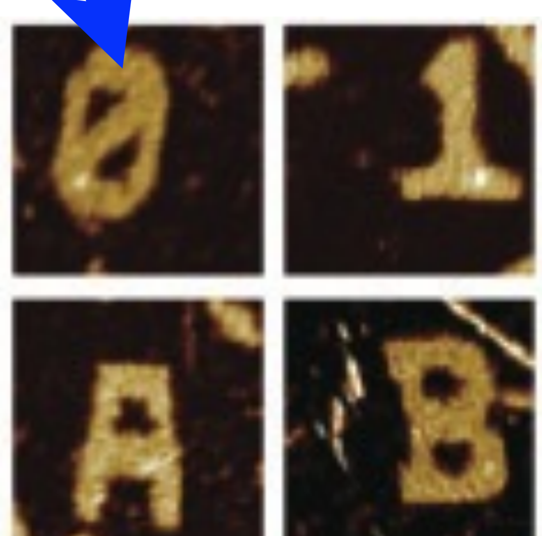
DNA origami 3D wireframe shapes
Benson, et al. Nature 2015

Nanostructure design via self-assembly

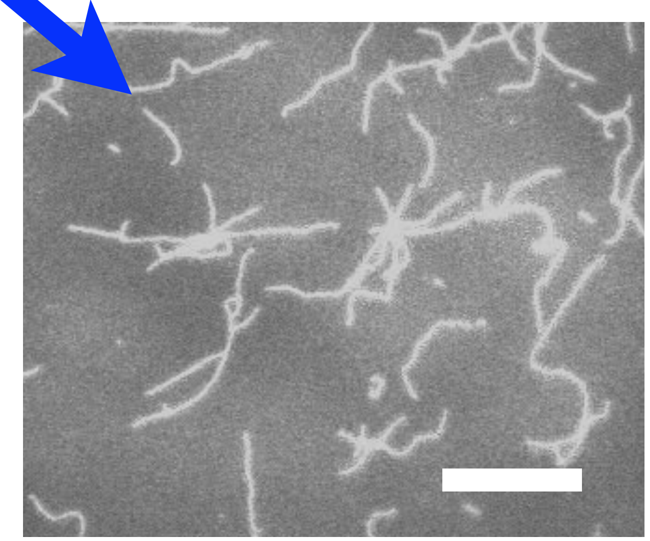
We tell the molecules **exactly** where to go

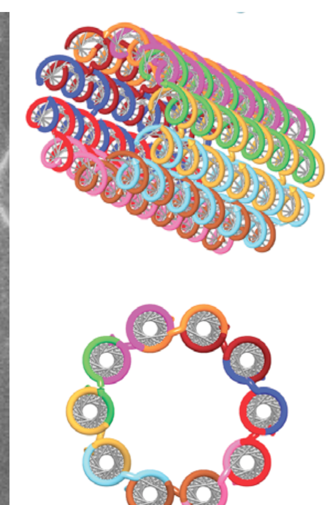


Rothemund
2006 Nature



Wei, Dai, Yin. 2013
Nature



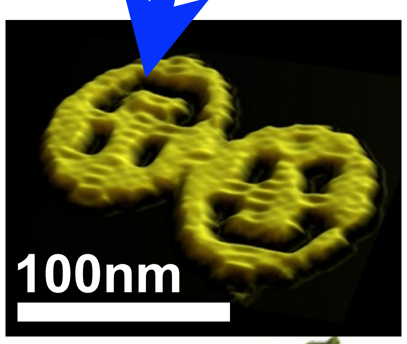


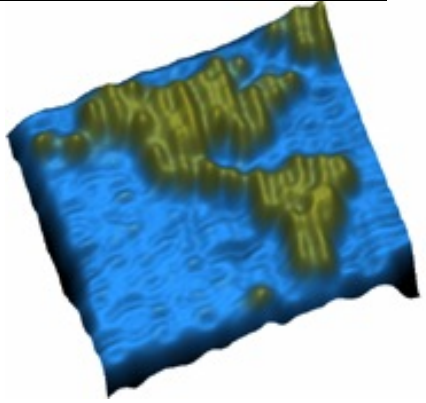
Yin et al 2008 Science

Nanostructure design via self-assembly

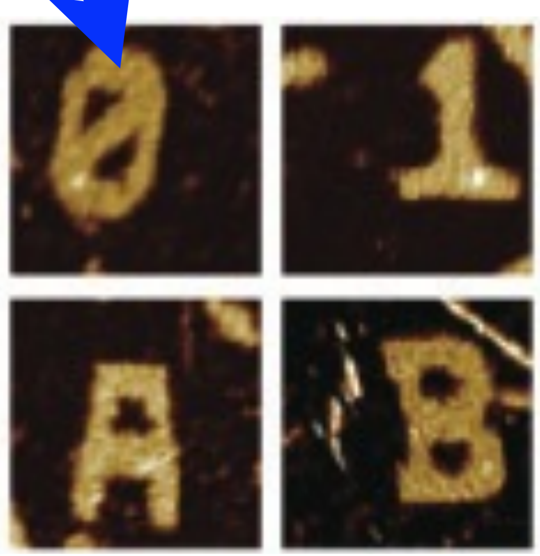
We tell the molecules **exactly** where to go

Can we have **smarter** molecules that decide where to go for themselves?

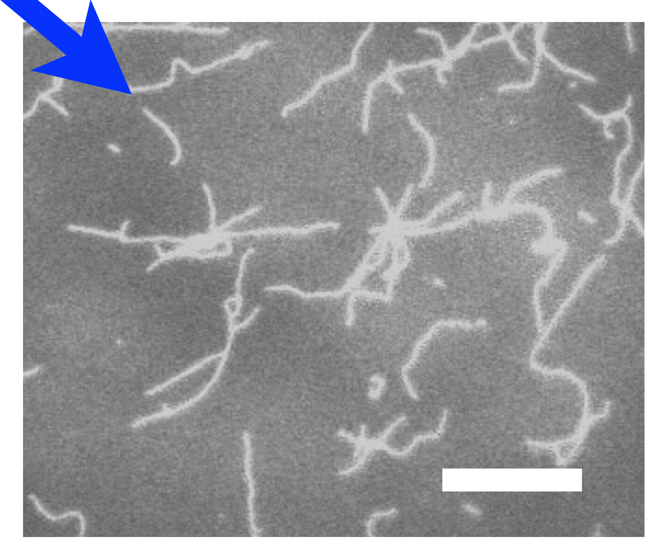


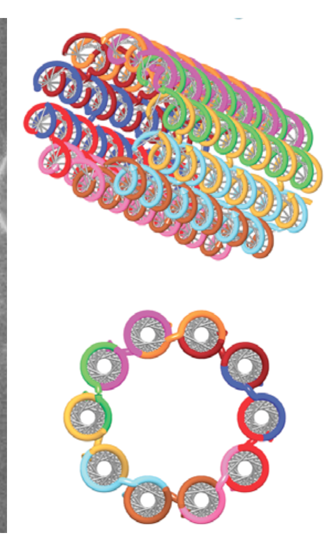


Rothemund 2006 Nature



Wei, Dai, Yin. 2013 Nature



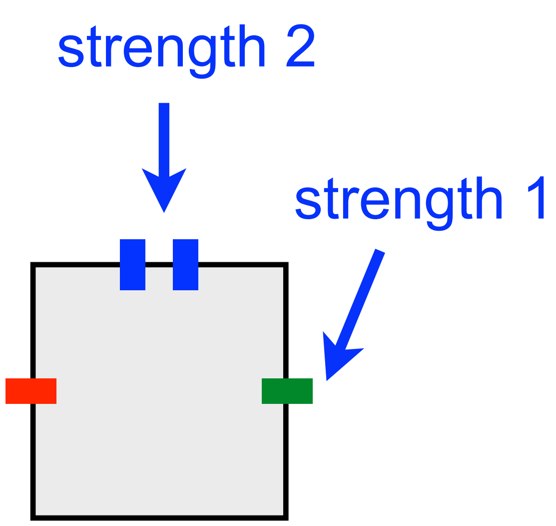


Yin et al 2008 Science

Computing with the abstract tile assembly model

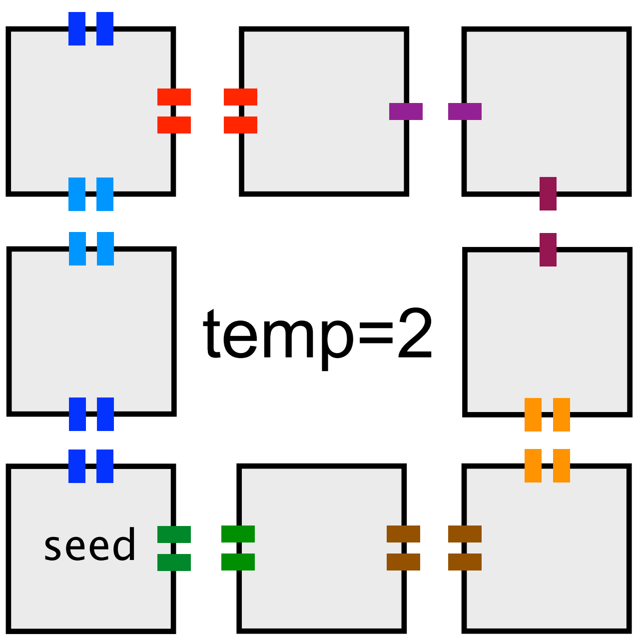
An asynchronous cellular automaton model capturing dynamics of molecular binding

- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998

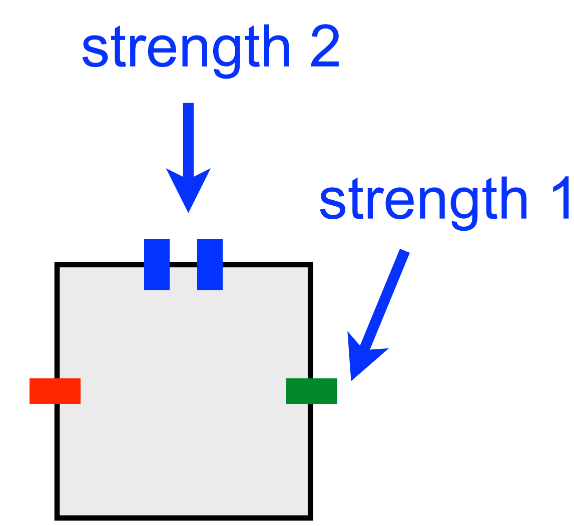


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

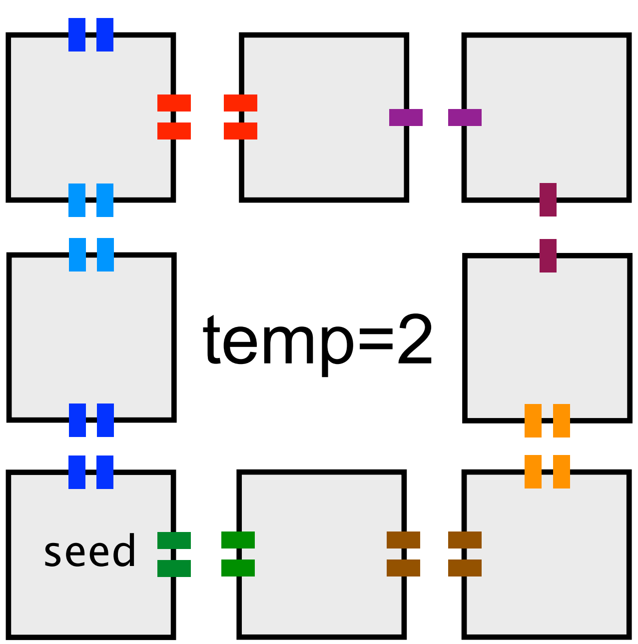
An asynchronous cellular automaton model capturing dynamics of molecular binding

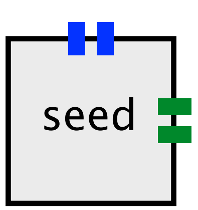
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



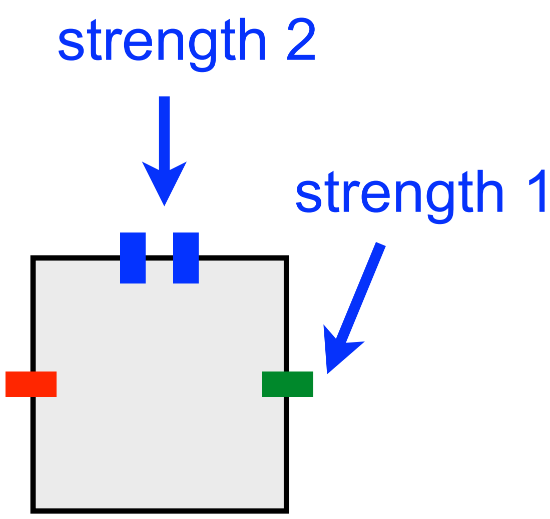


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

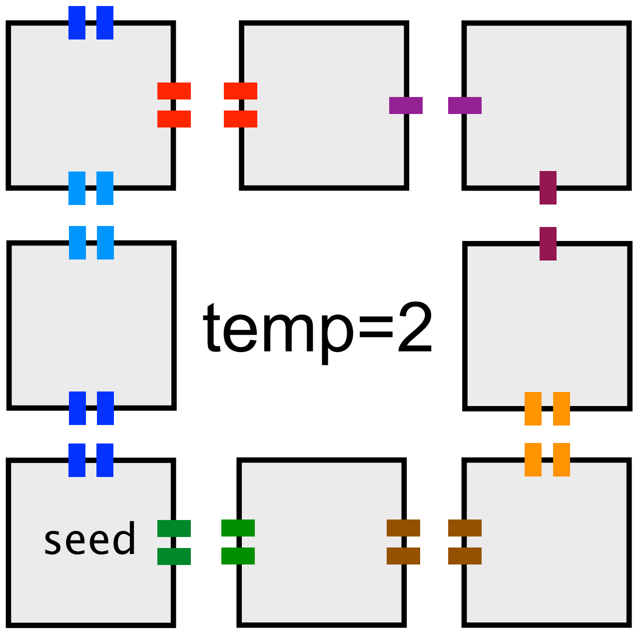
An asynchronous cellular automaton model capturing dynamics of molecular binding

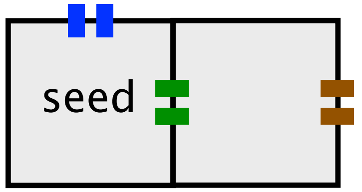
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



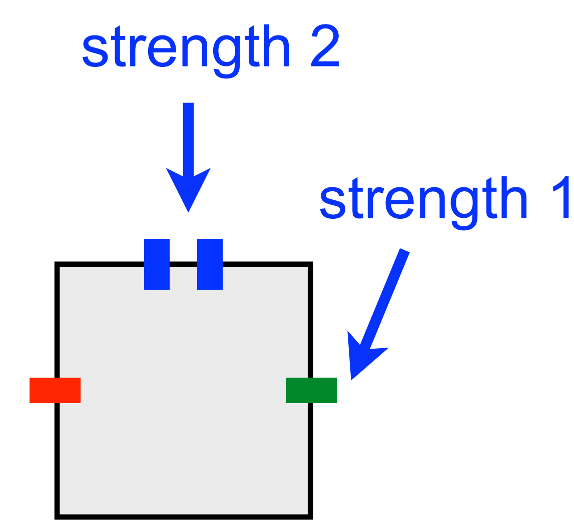


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

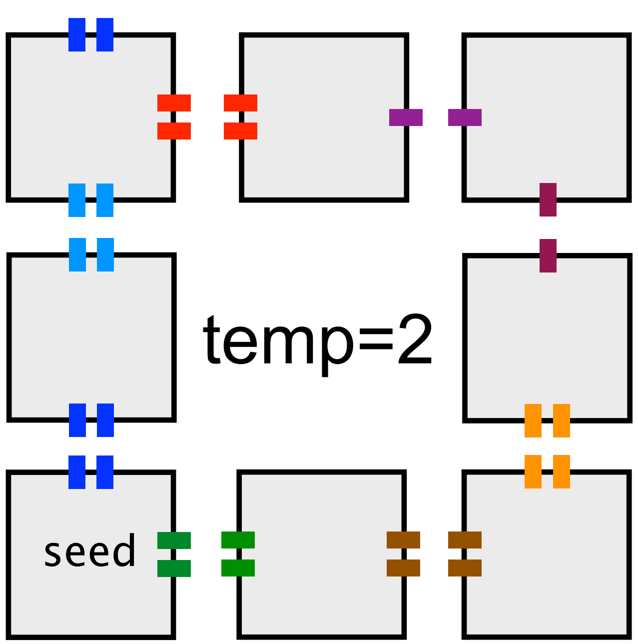
An asynchronous cellular automaton model capturing dynamics of molecular binding

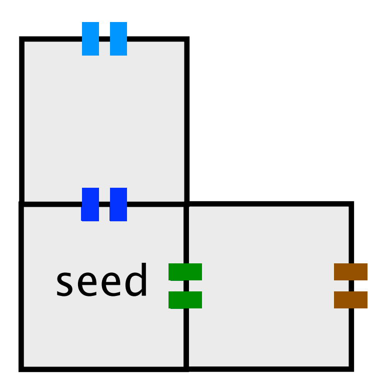
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



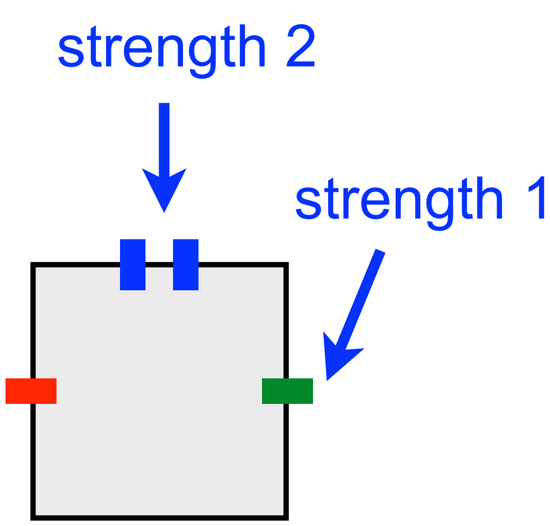


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

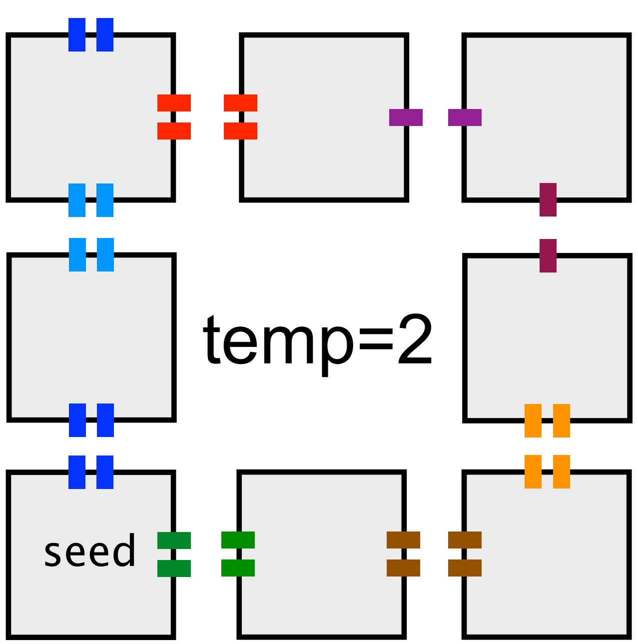
An asynchronous cellular automaton model capturing dynamics of molecular binding

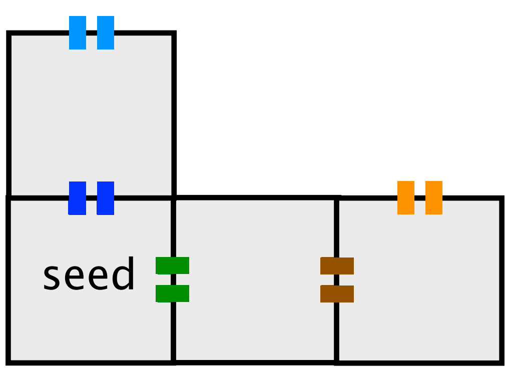
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



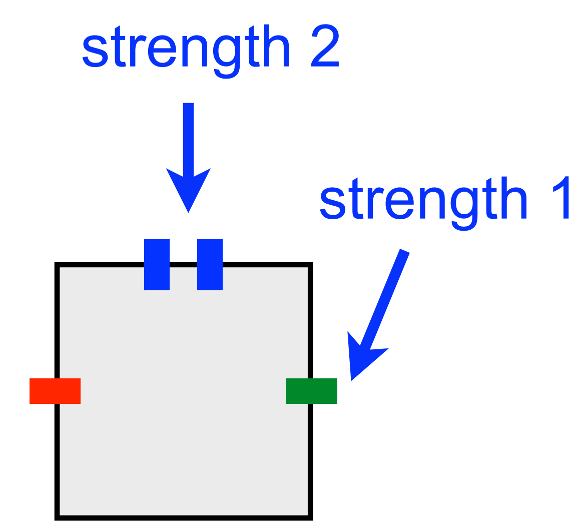


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

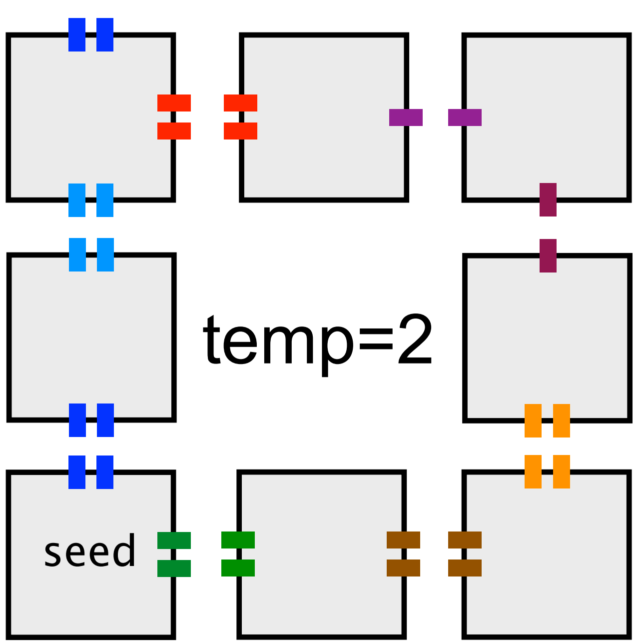
An asynchronous cellular automaton model capturing dynamics of molecular binding

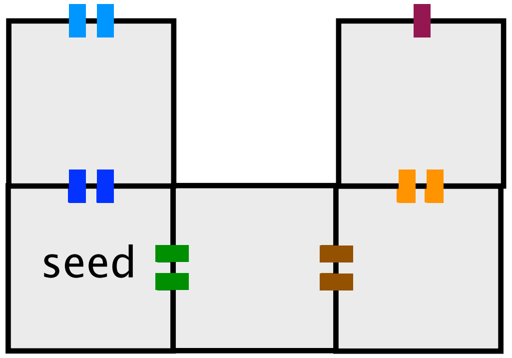
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



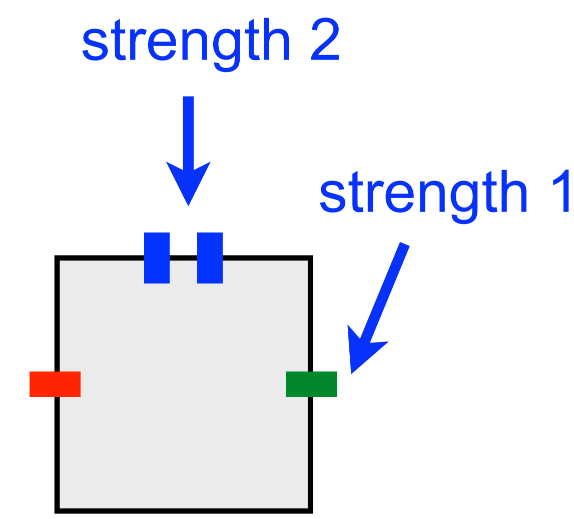


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

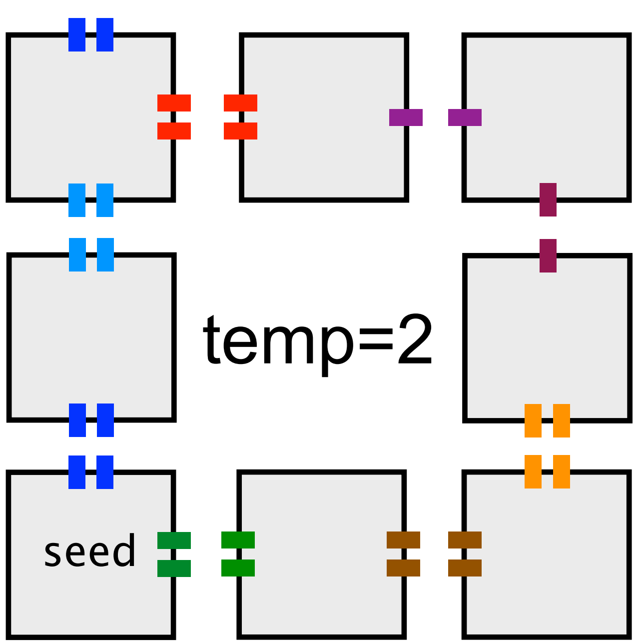
An asynchronous cellular automaton model capturing dynamics of molecular binding

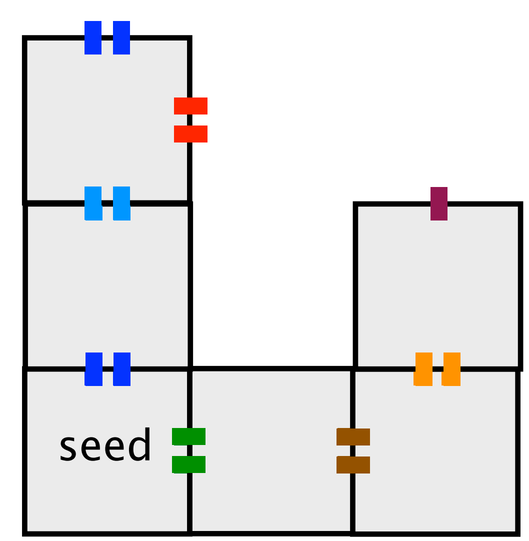
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



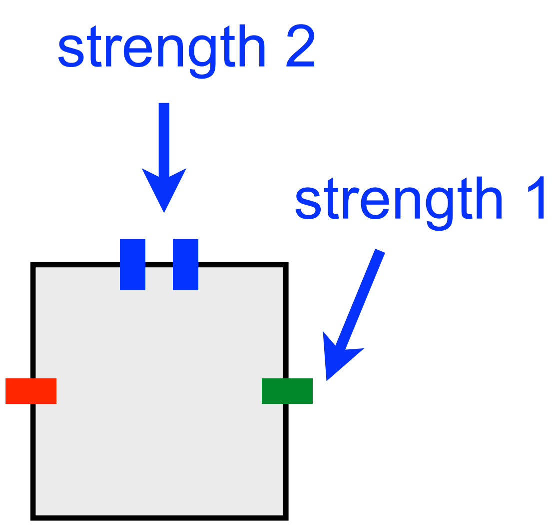


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

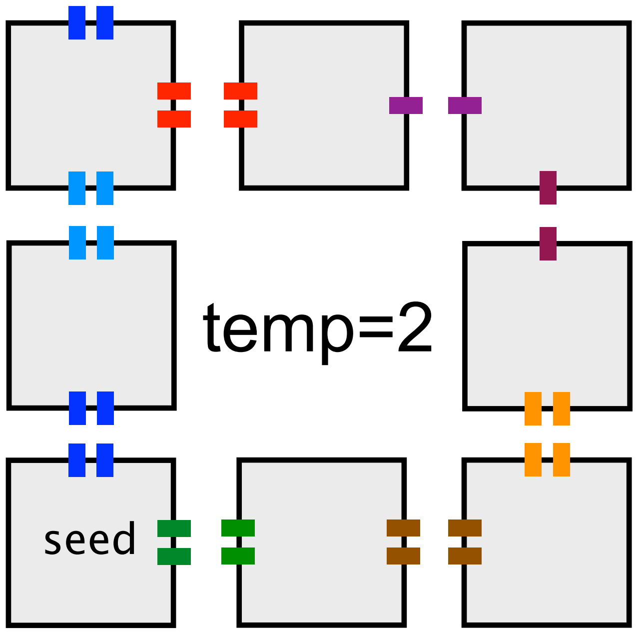
An asynchronous cellular automaton model capturing dynamics of molecular binding

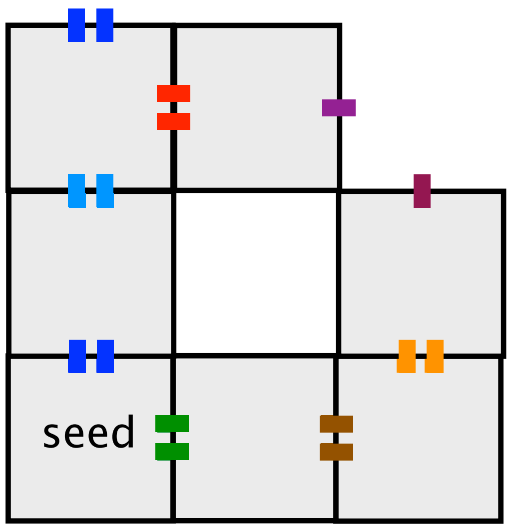
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

Model by Winfree, 1998



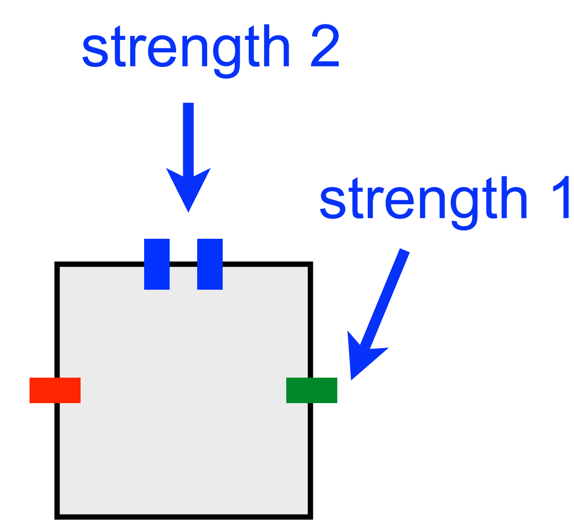


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

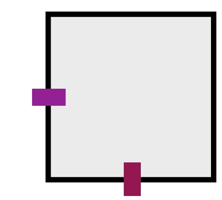
Computing with the abstract tile assembly model

An asynchronous cellular automaton model capturing dynamics of molecular binding

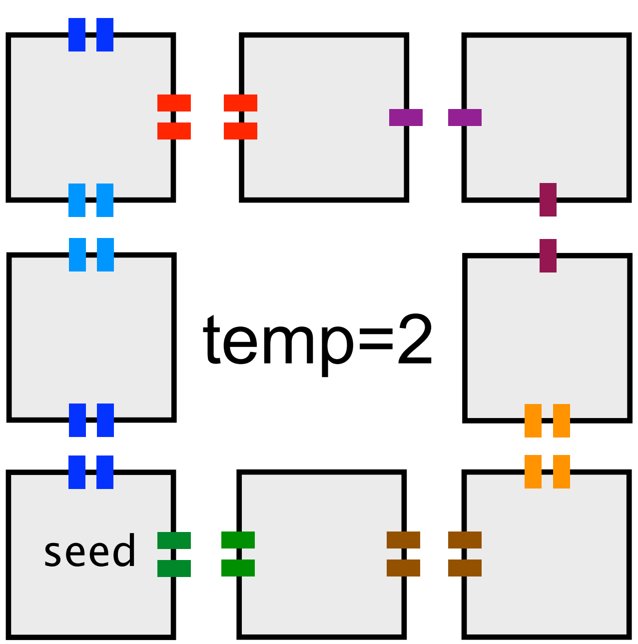
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)

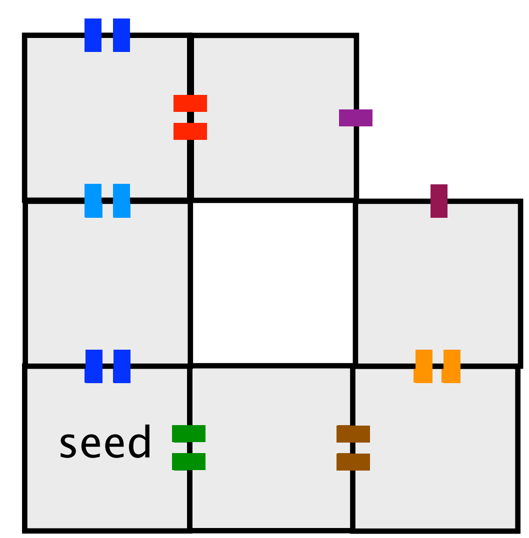


- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature



Model by Winfree, 1998



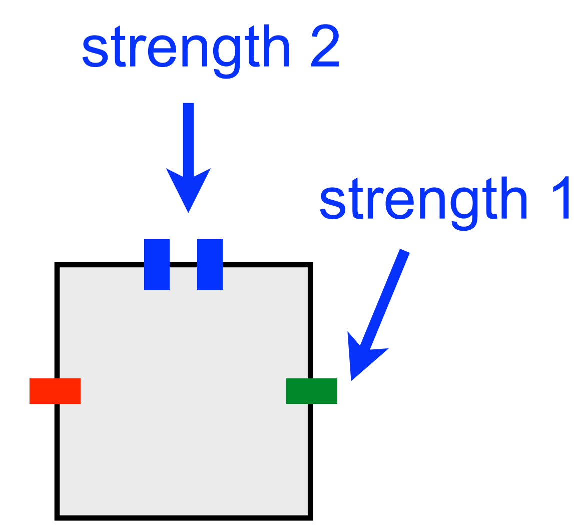


We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Computing with the abstract tile assembly model

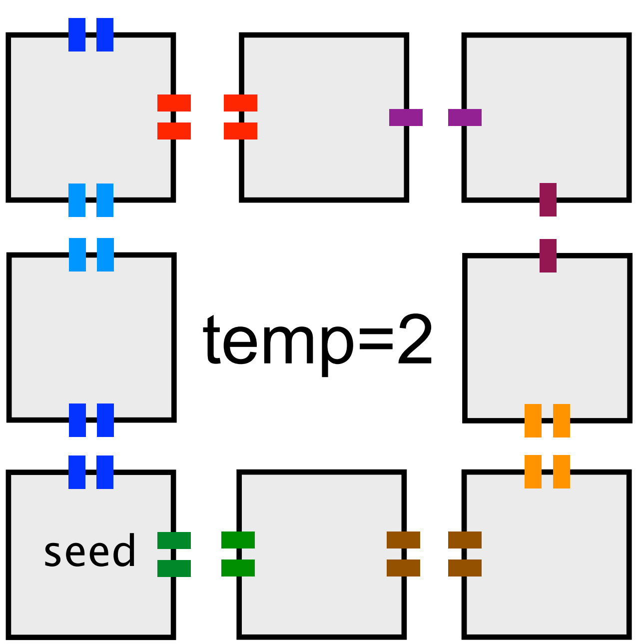
An asynchronous cellular automaton model capturing dynamics of molecular binding

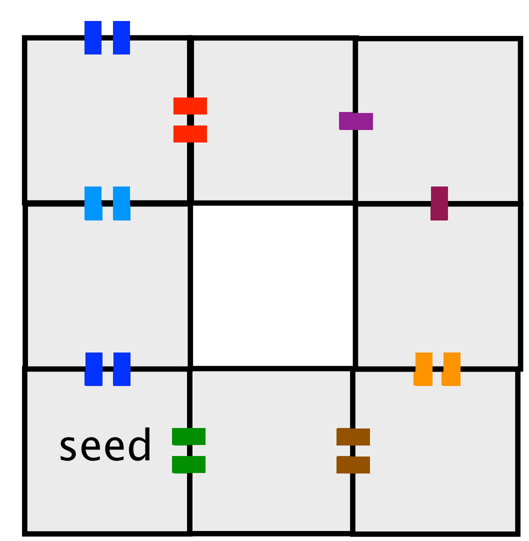
- Square **tiles**
  - finite set of tile types, unlimited supply of each type, non-rotatable
- Each side has a **glue** (colour) and **strength** (0,1,2,3,…)
- System has a **temperature** (e.g. 2)



- **Simple local binding rule**: A tile sticks to an assembly if enough of its glues match so that the sum of the strengths of the matching glues is at least the temperature

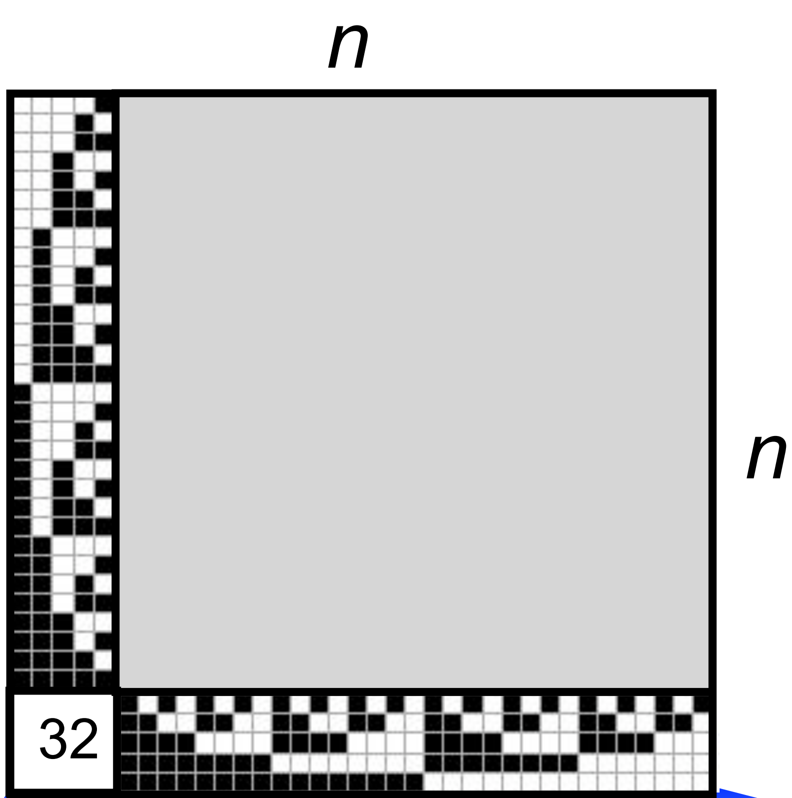
Model by Winfree, 1998





We can make these tiles out of DNA!
Small size, requires us to program in a bottom-up way

Algorithmic self-assembly **theory**: previous work



- Turing universality

  Winfree, PhD Thesis. 1998

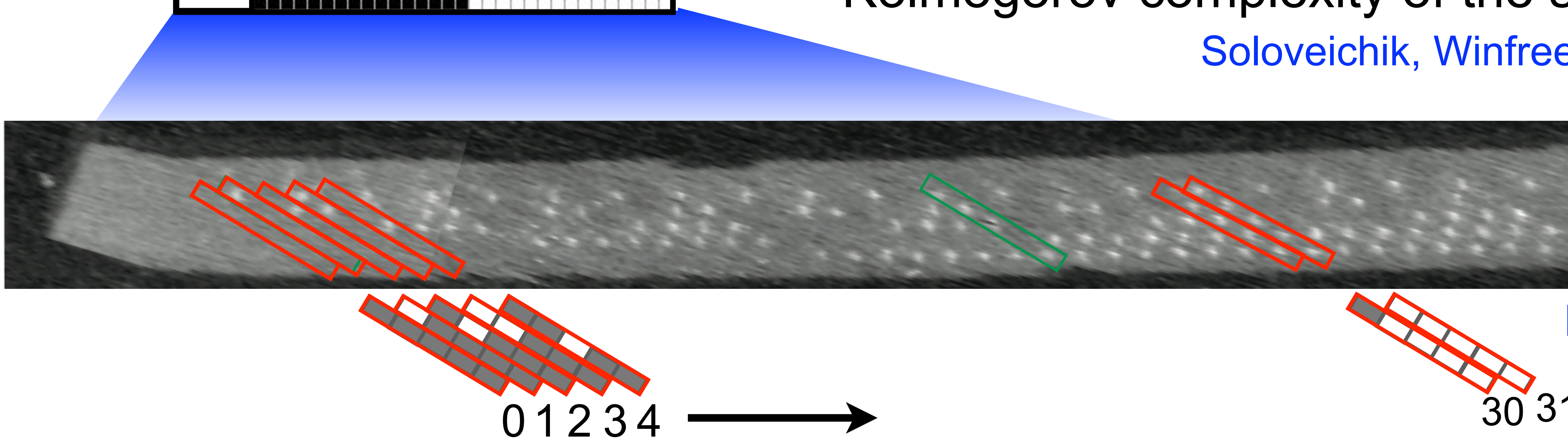
- Efficient assembly of simple shapes: *n* x *n* squares using Θ(log *n* / log log *n*) tile types

  Adleman, Cheng, Goel, Huang STOC 2001

  Rothemund, Winfree. STOC 2000

- Efficient assembly of scaled shapes using a number of tile types roughly equal to the Kolmogorov complexity of the shape

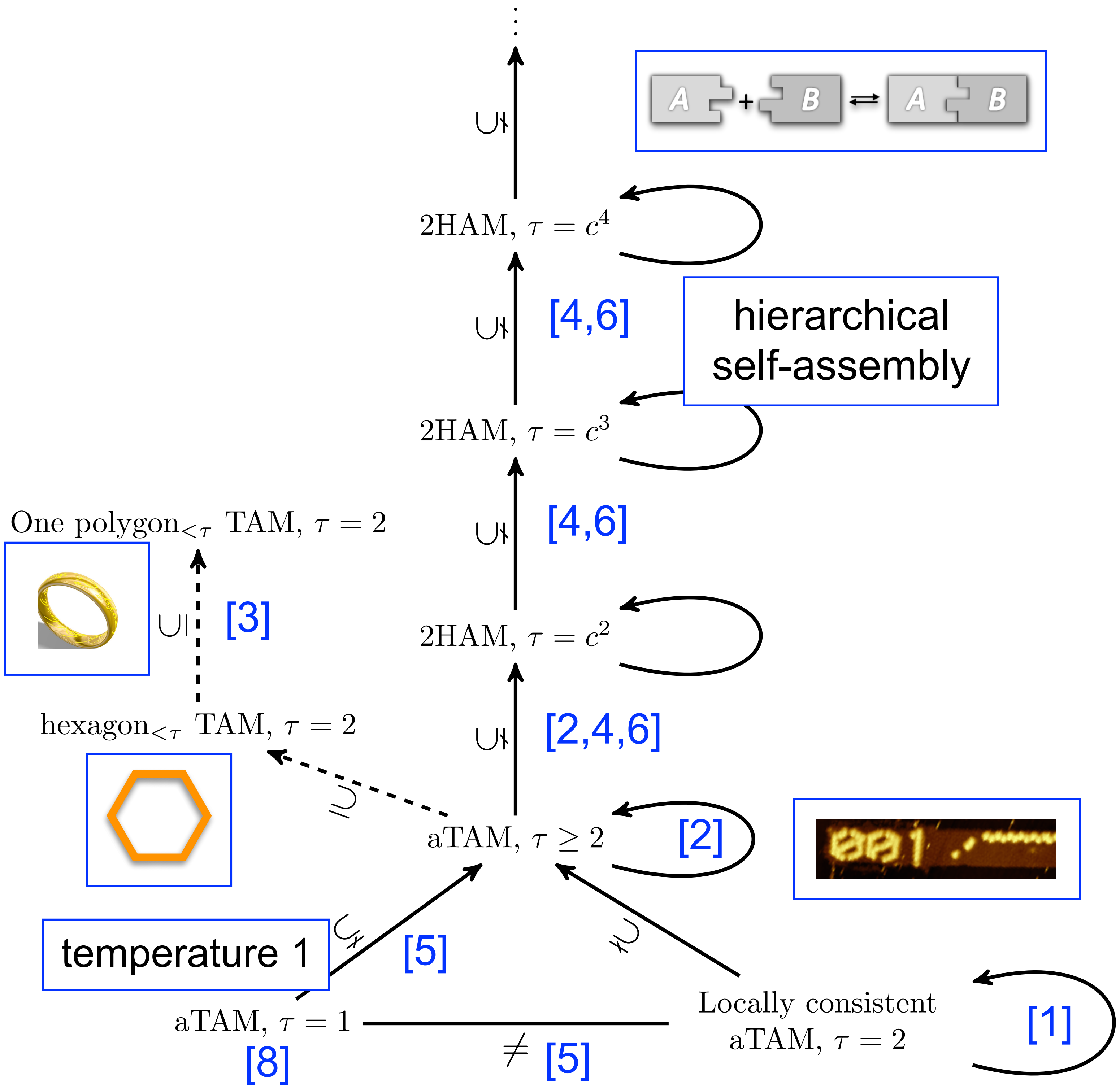
  Soloveichik, Winfree. SICOMP 2007



Evans (Winfree group) PhD Thesis 2014

Algorithmic self-assembly **theory**: previous work

Simulation & *intrinsic* universality

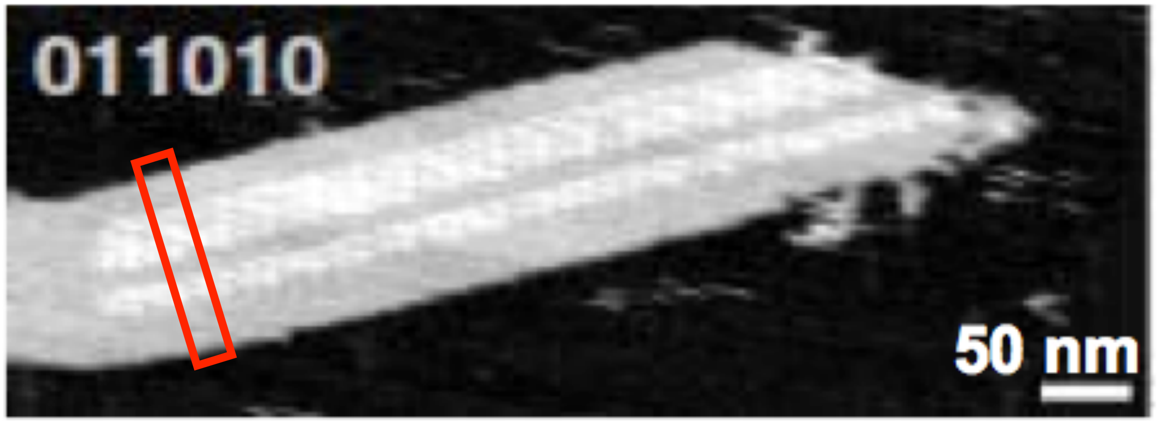


Complexity hierarchy for self-assembly

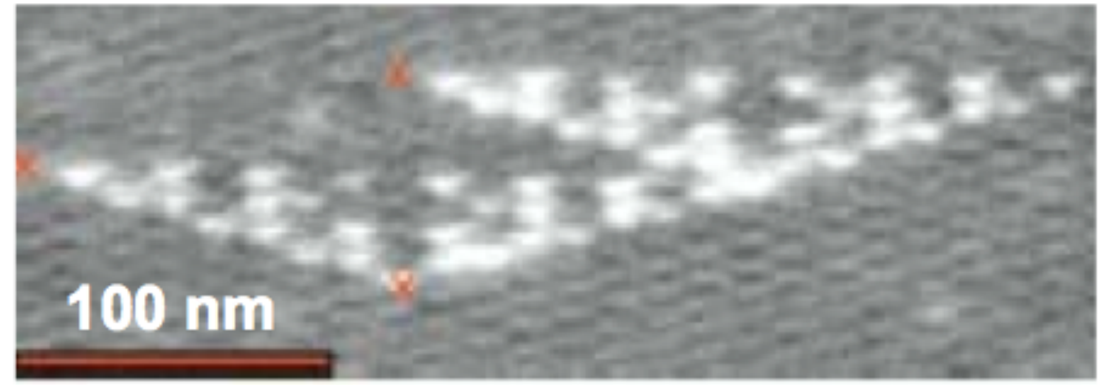
Characterisations and comparisons of self-assembly models based on **simulation** between self-assembly systems

[8] STOC 2017
[7] Phil Trans Royal Soc. A. 2015
[6] Algorithmica 2015
[5] SODA 2014
[4] ICALP 2014
[3] ICALP 2013
[2] FOCS 2012
[1] STACS 2012

Algorithmic self-assembly **experiments**: previous work

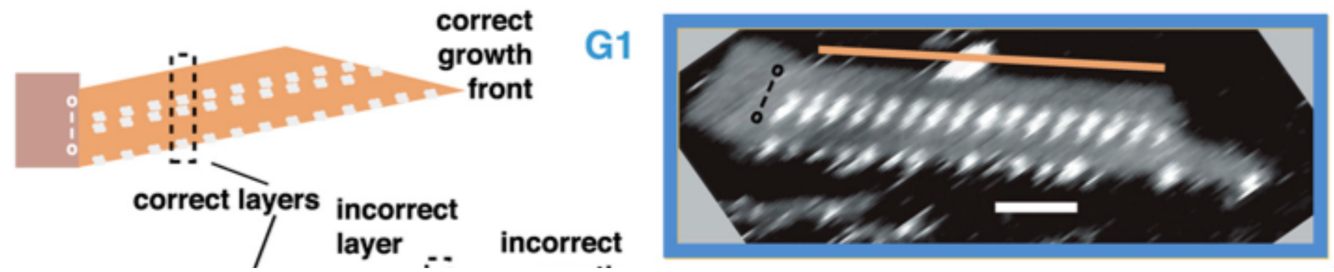


Bit Copying. Barish et al.2009

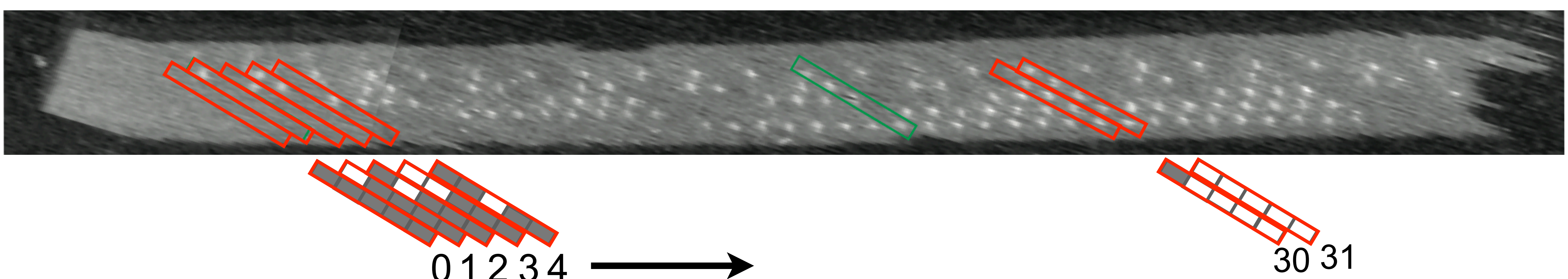


Sierpinski Triangles. Rothemund, Papadakis, Winfree. 2004

Counter. Barish et al. 2009



Copying & replication Schulman, Yurke, Winfree. PNAS. 2012



Evans, Winfree group. PhD Thesis 2014

Copying, Sierpinsky, binary counting to 31: Can we run more algorithms?

Structure of talk

Copying, Sierpinski, binary counting to 31, can we run more algorithms?

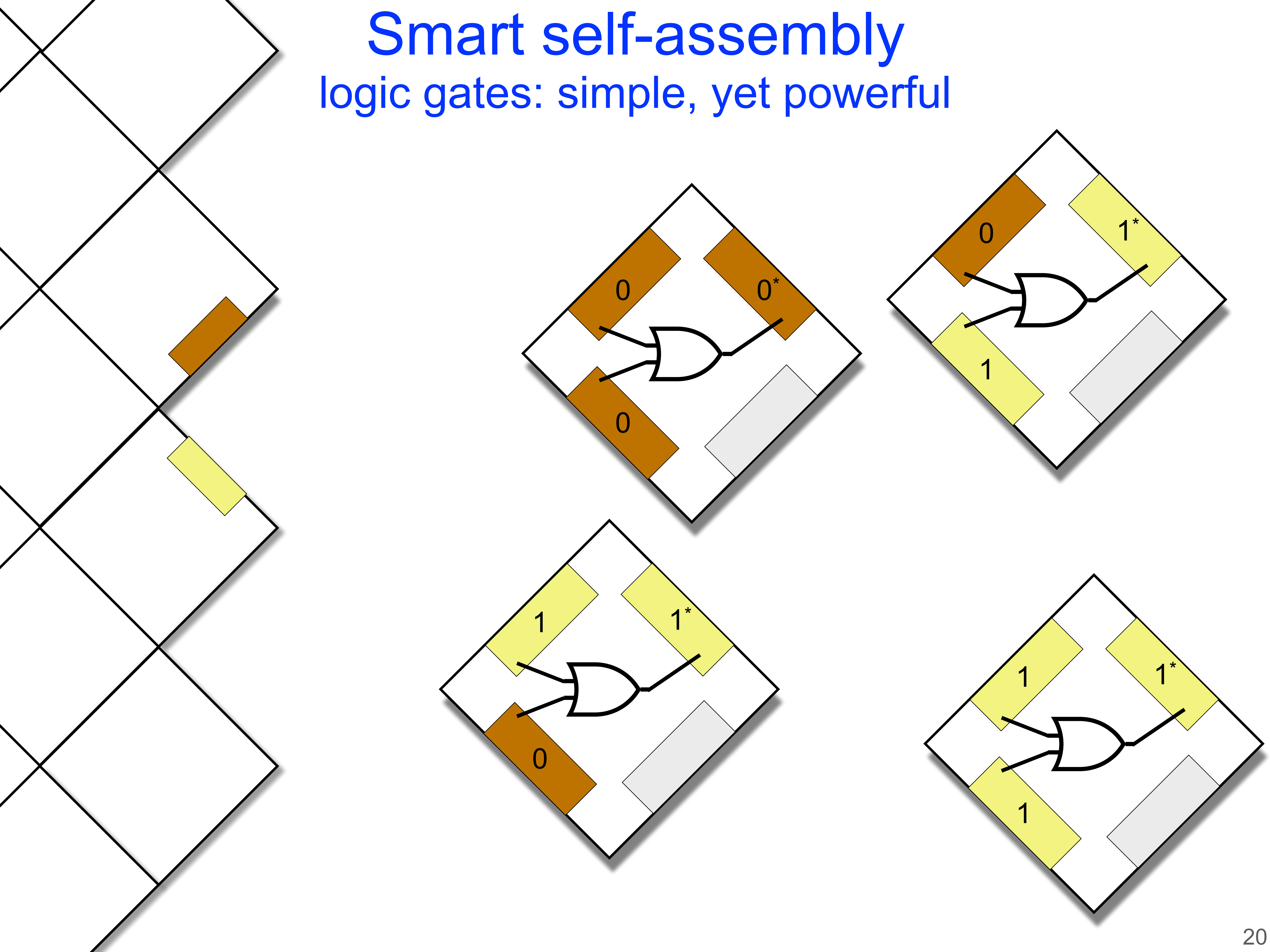
Theoretical circuit model

How it works: design and implementation

Experimental results

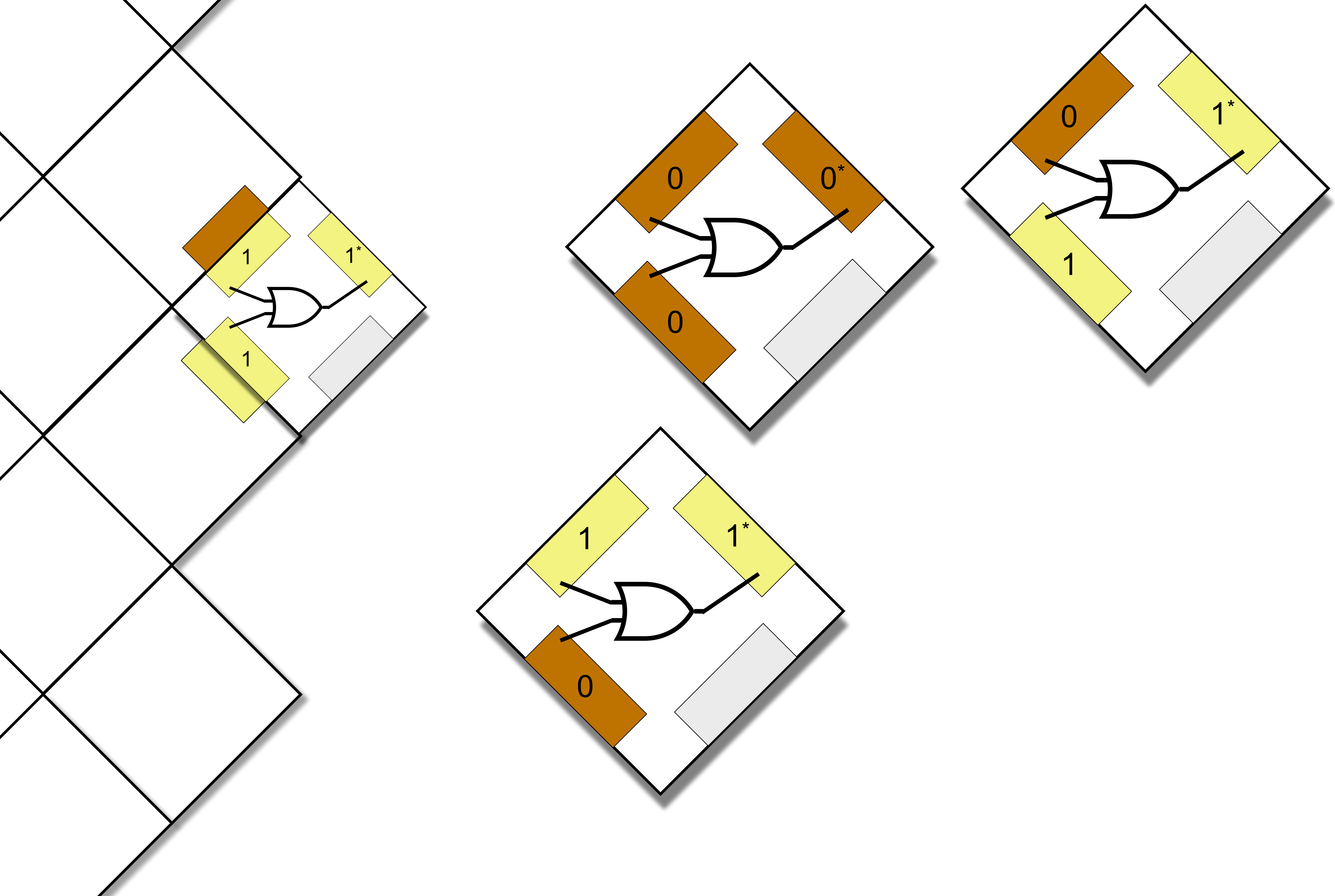
Smart self-assembly

logic gates: simple, yet powerful



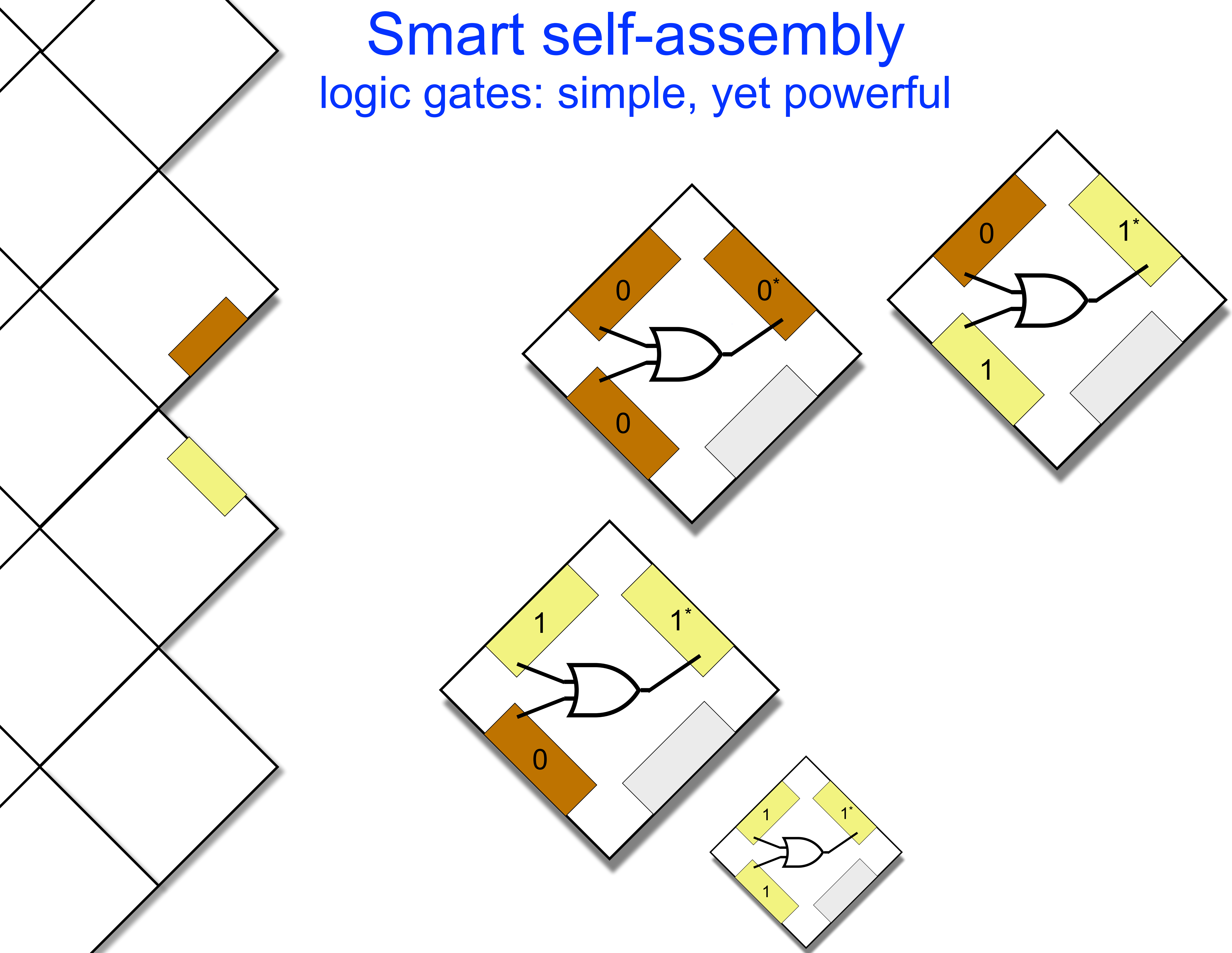
Smart self-assembly

logic gates: simple, yet powerful



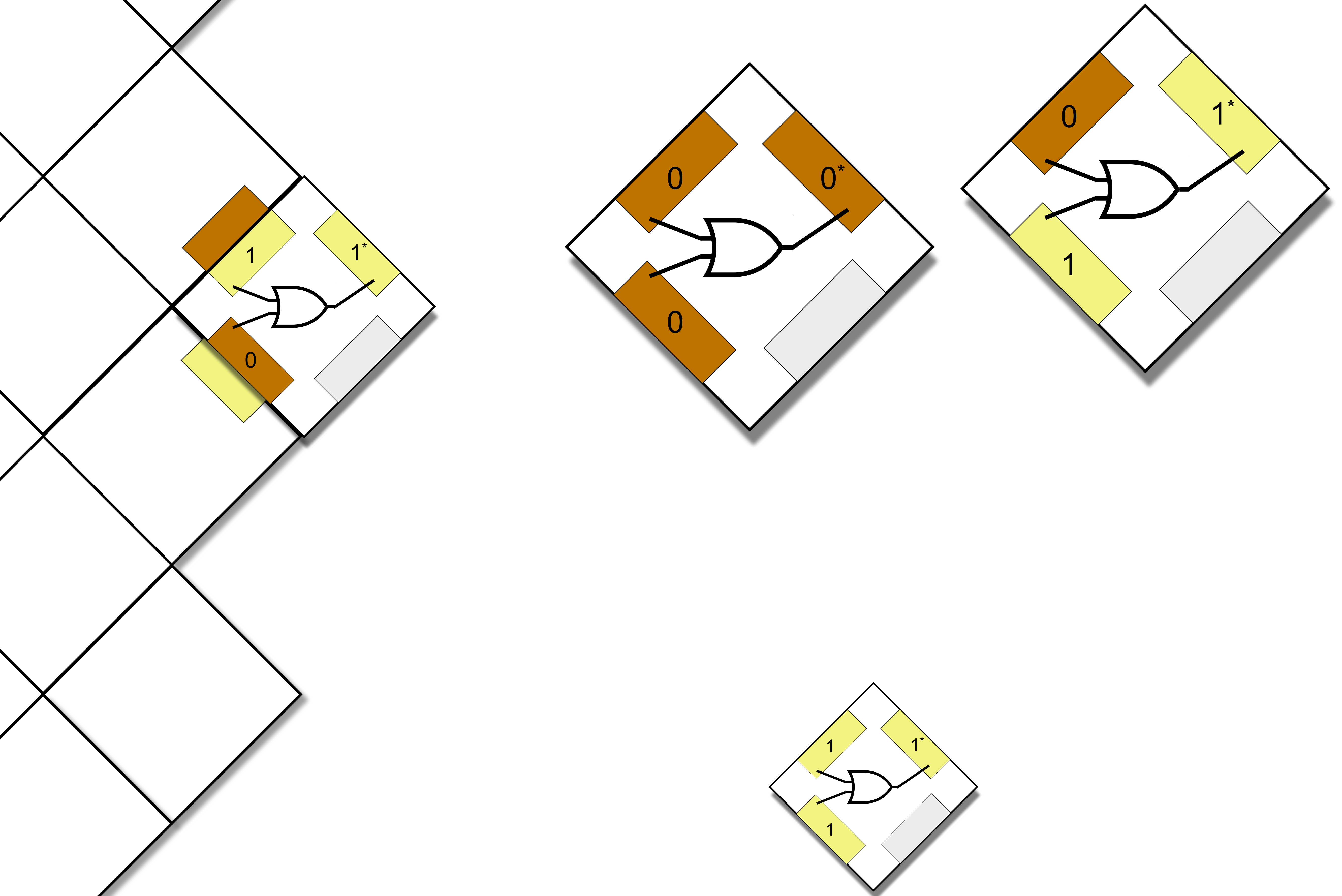
Smart self-assembly

logic gates: simple, yet powerful



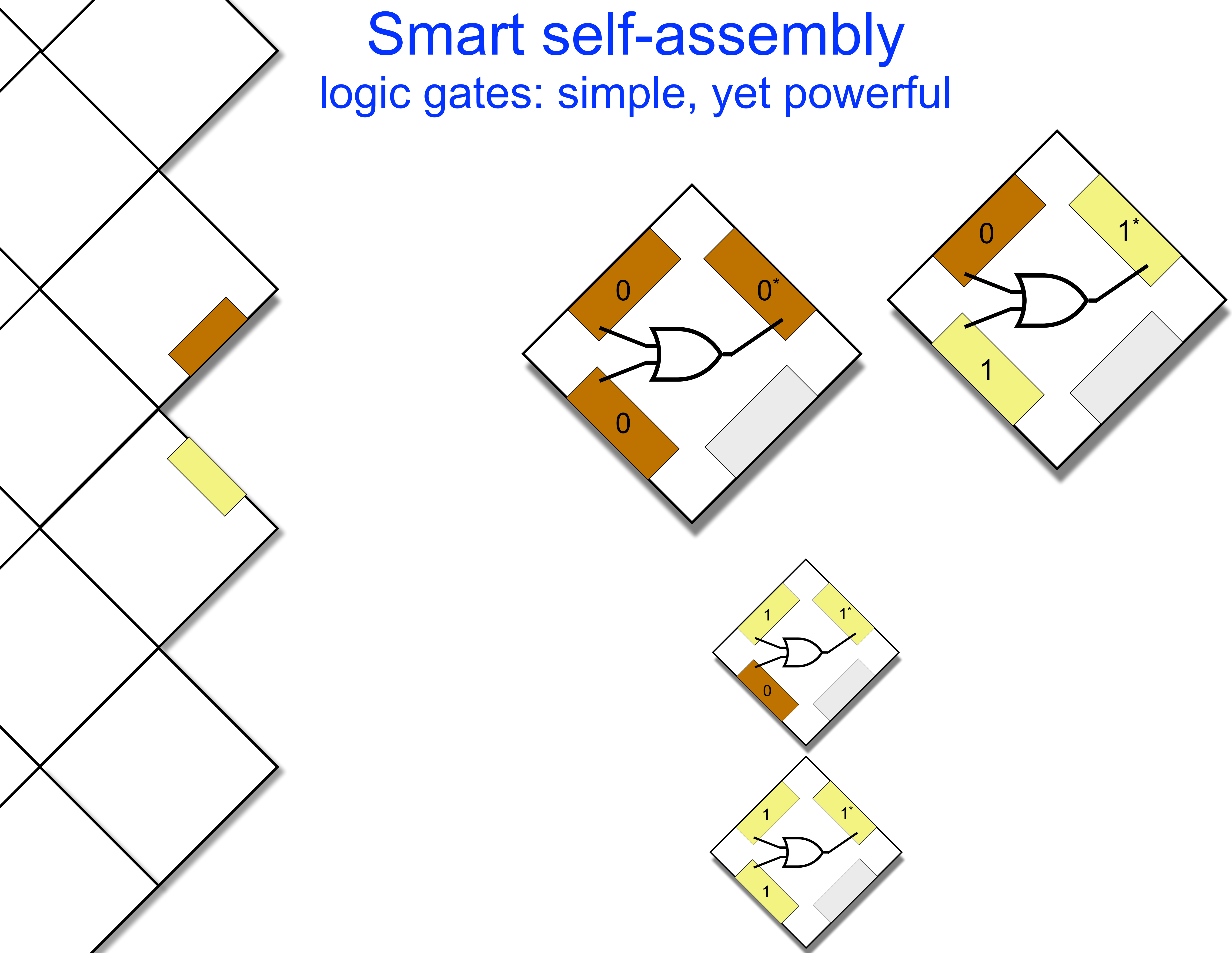
Smart self-assembly

logic gates: simple, yet powerful



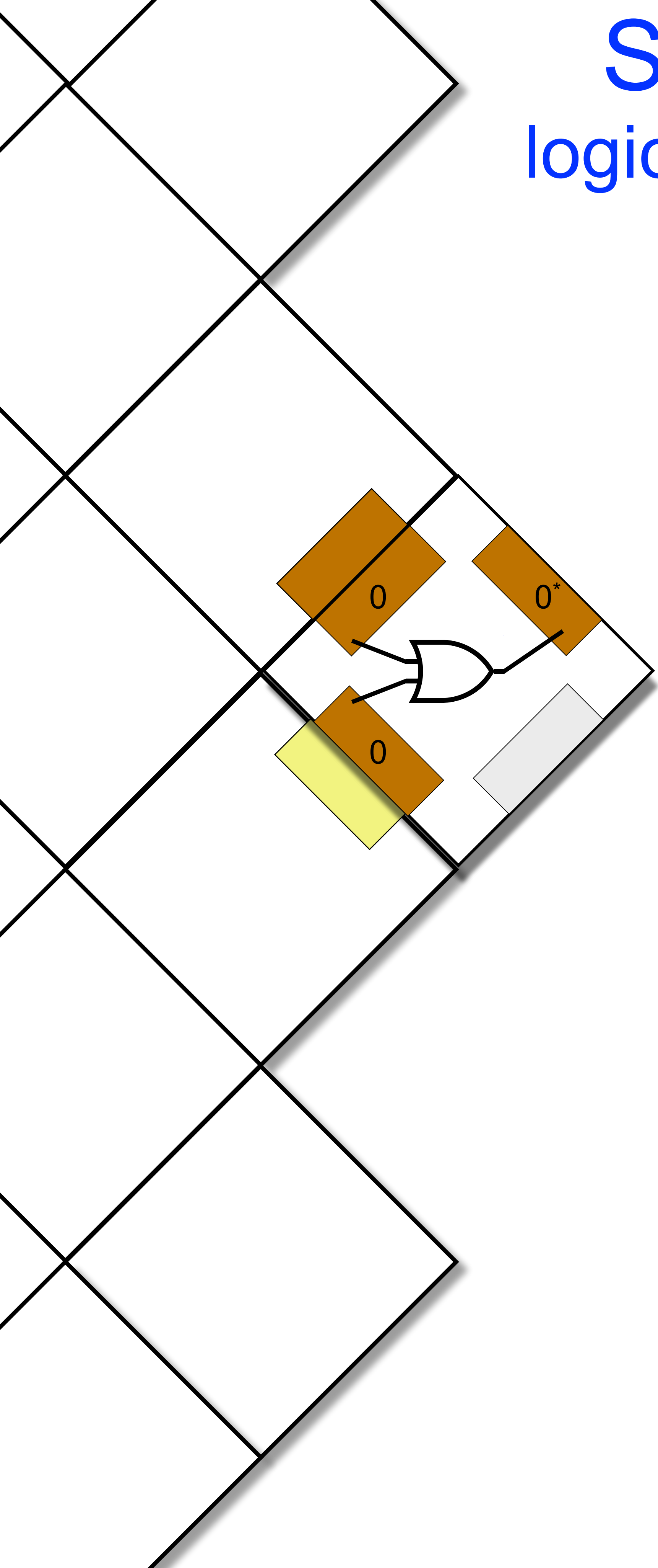
Smart self-assembly

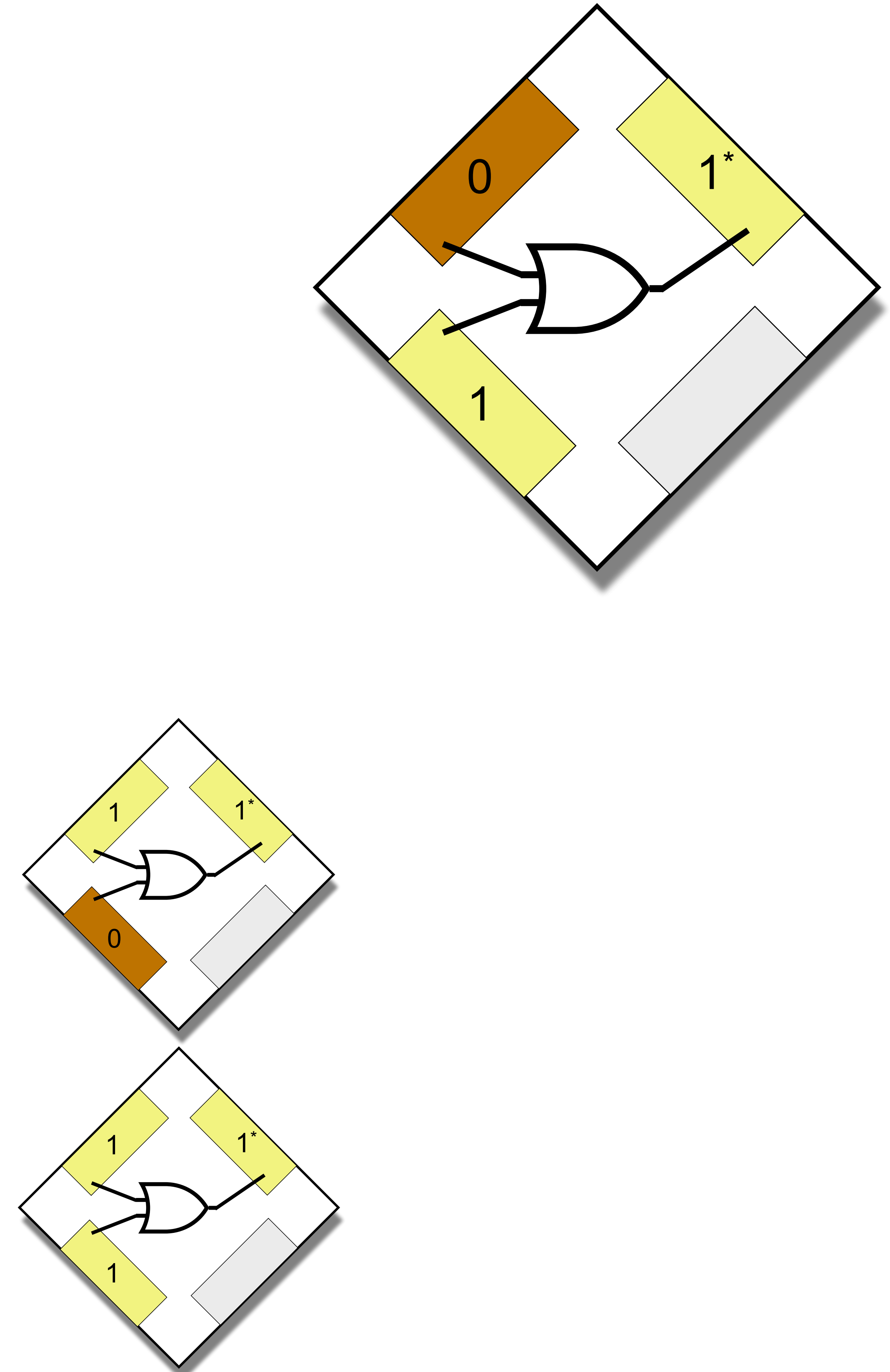
logic gates: simple, yet powerful



Smart self-assembly

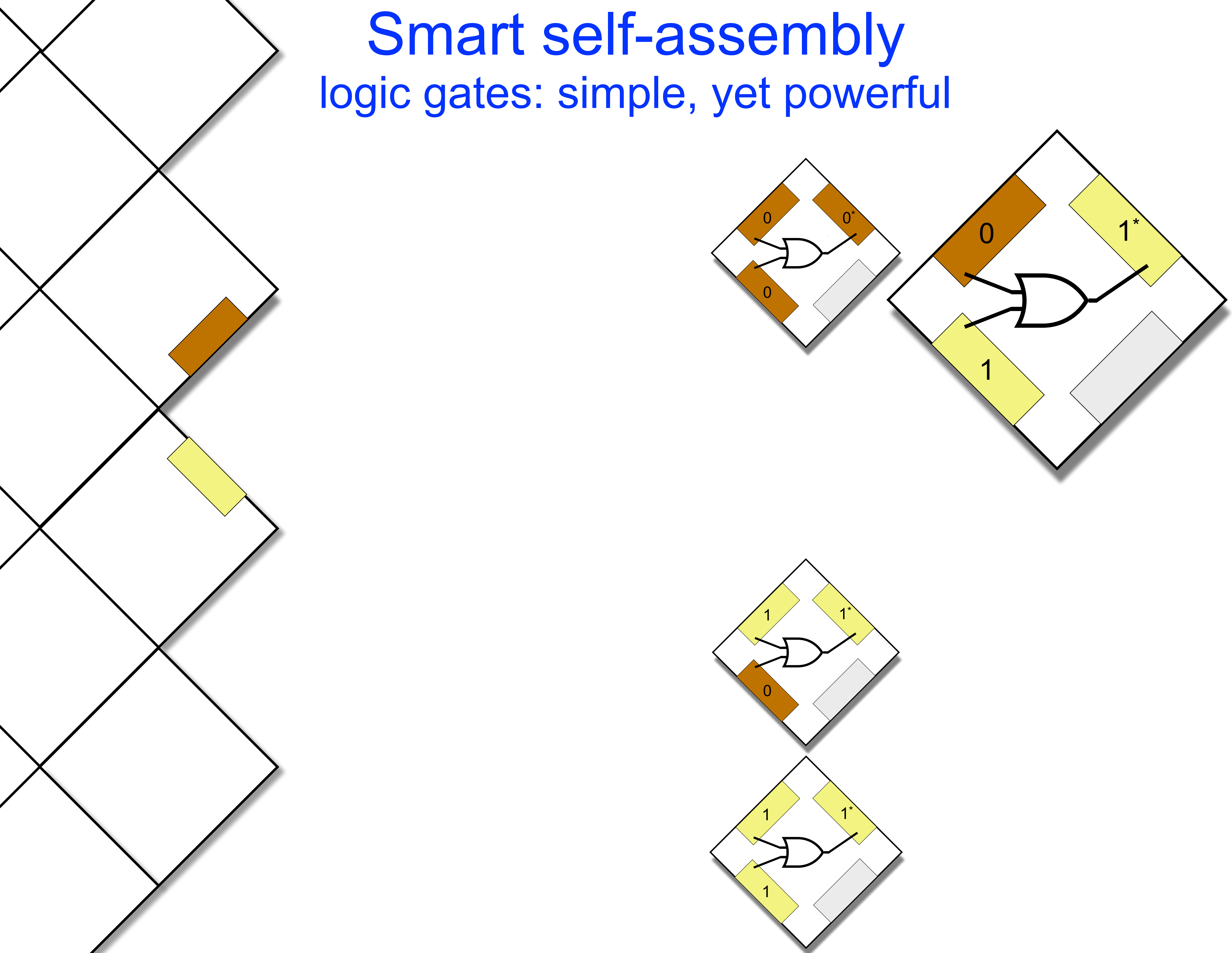
logic gates: simple, yet powerful





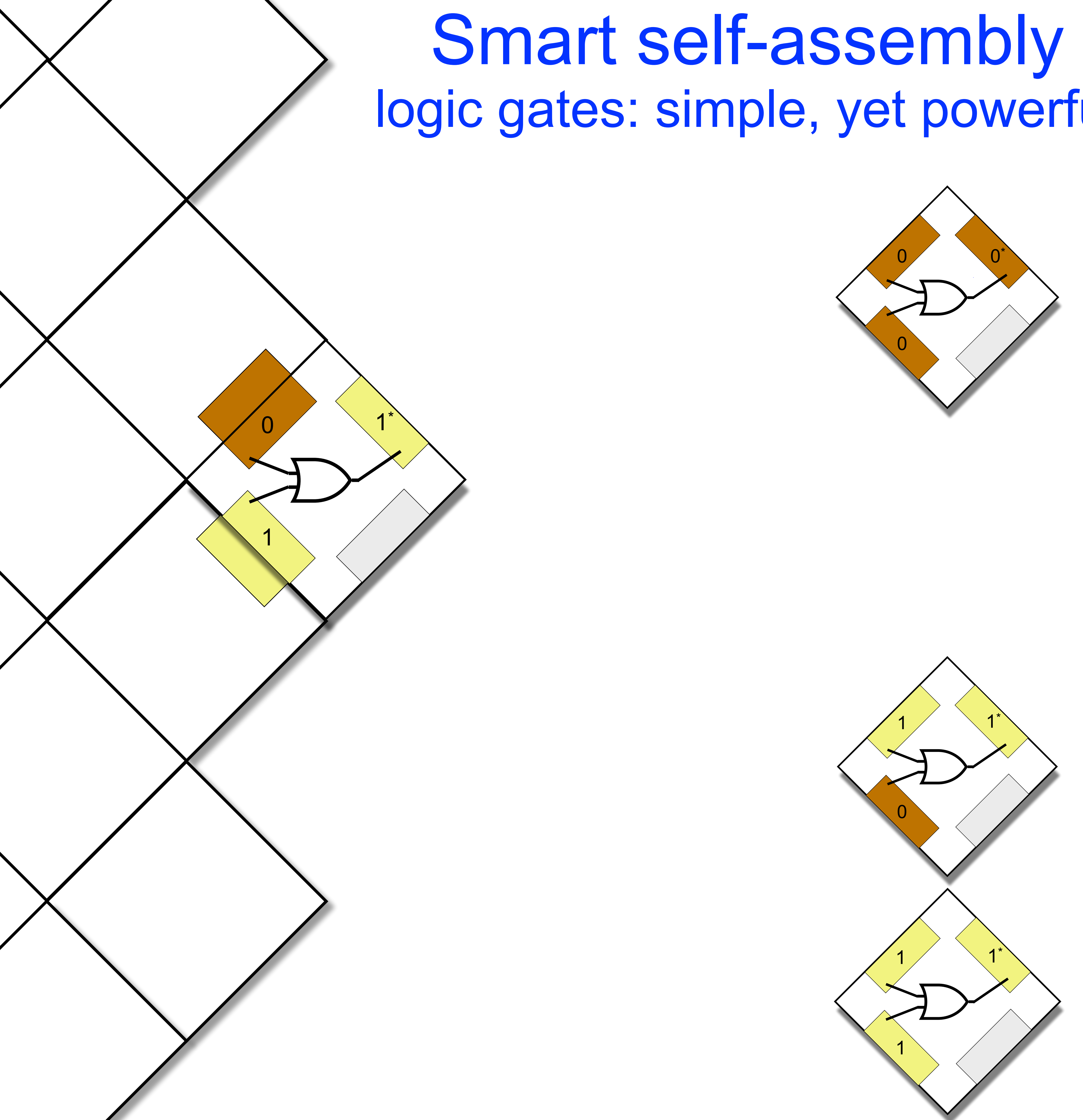
Smart self-assembly

logic gates: simple, yet powerful



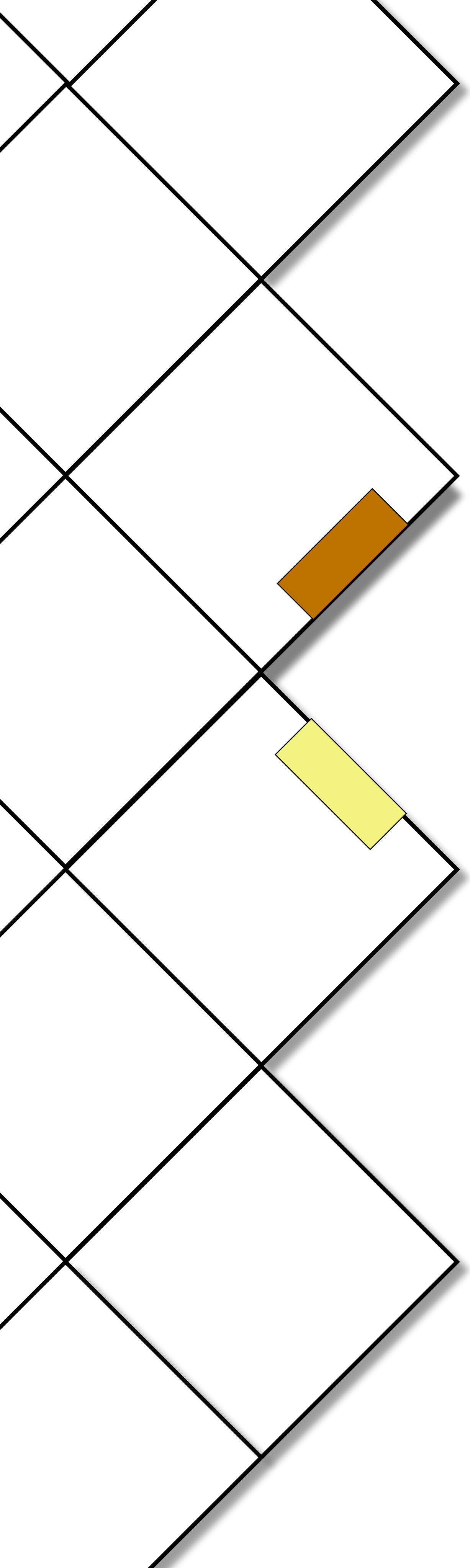
Smart self-assembly

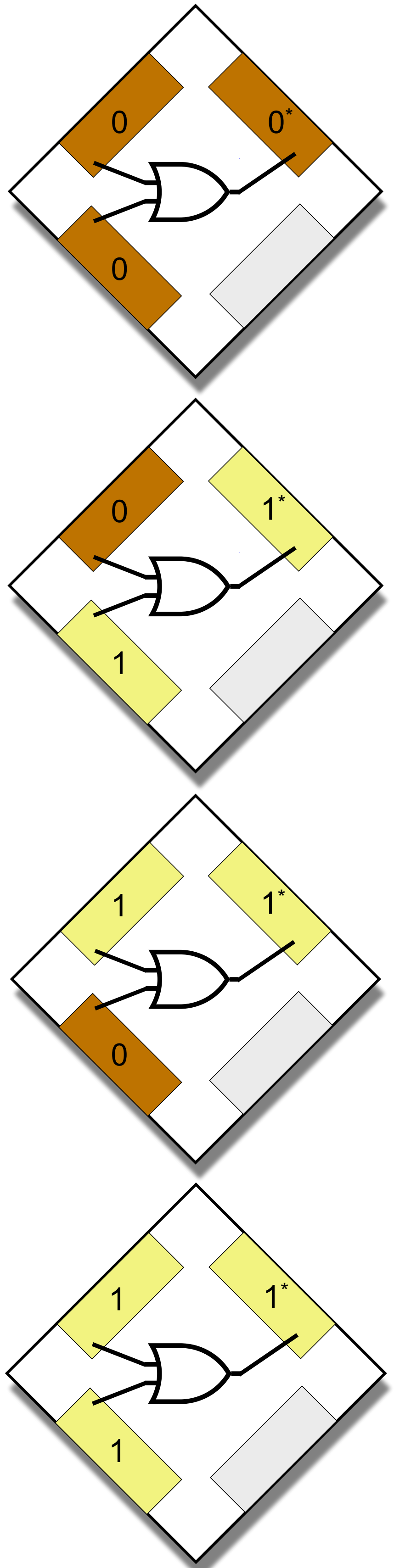
logic gates: simple, yet powerful



Smart self-assembly

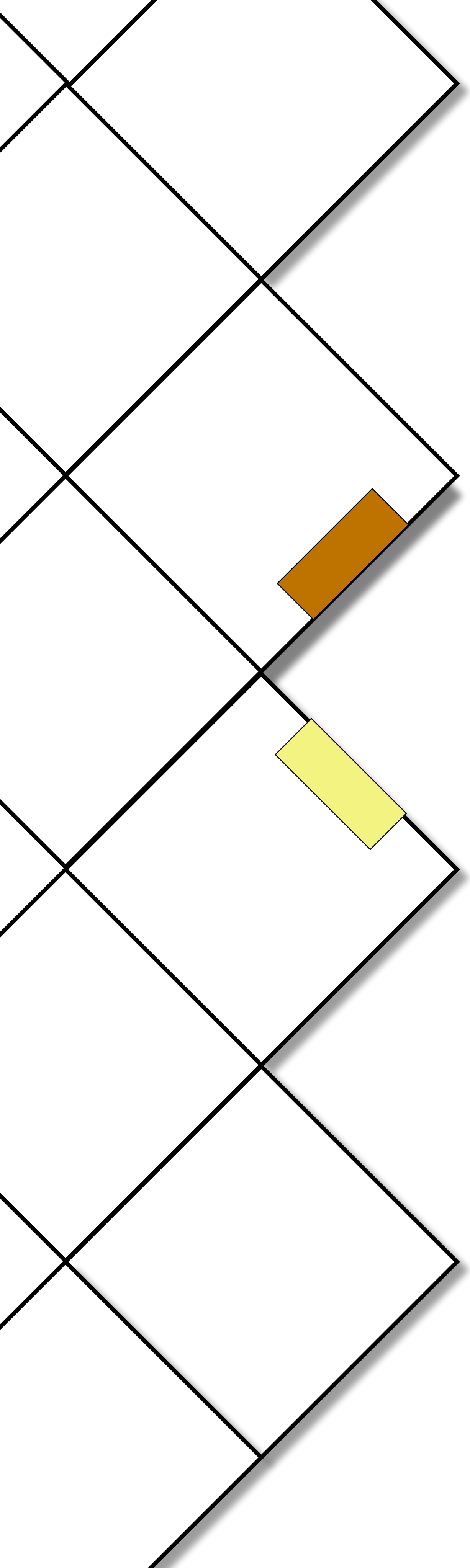
logic gates: simple, yet powerful

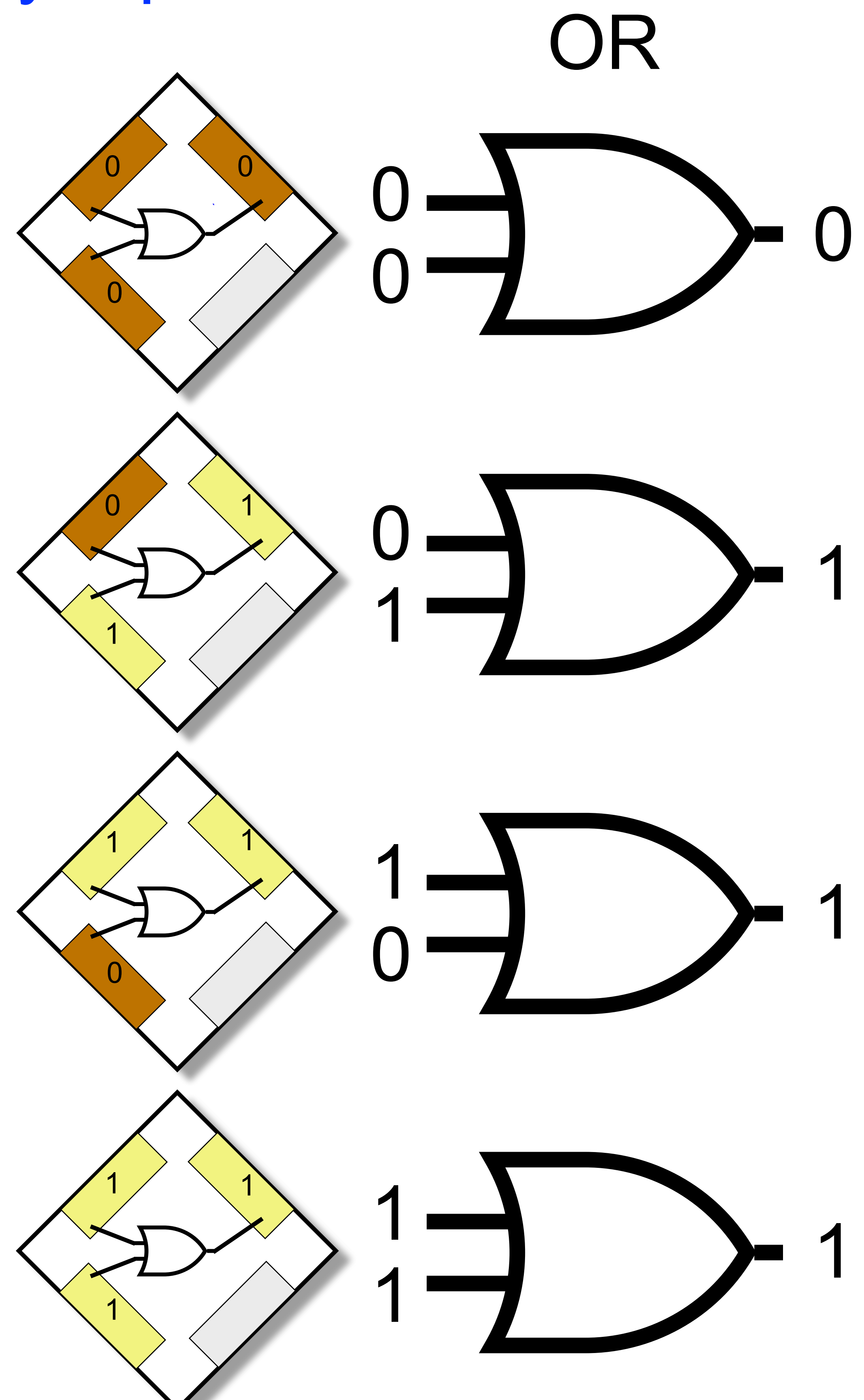




Smart self-assembly

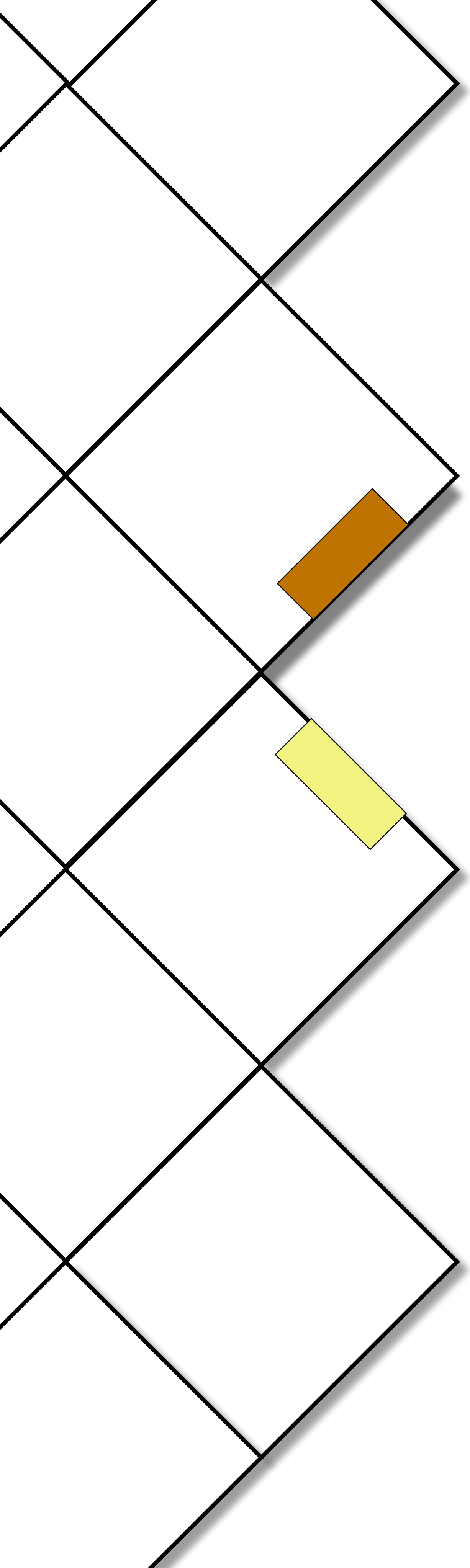
logic gates: simple, yet powerful

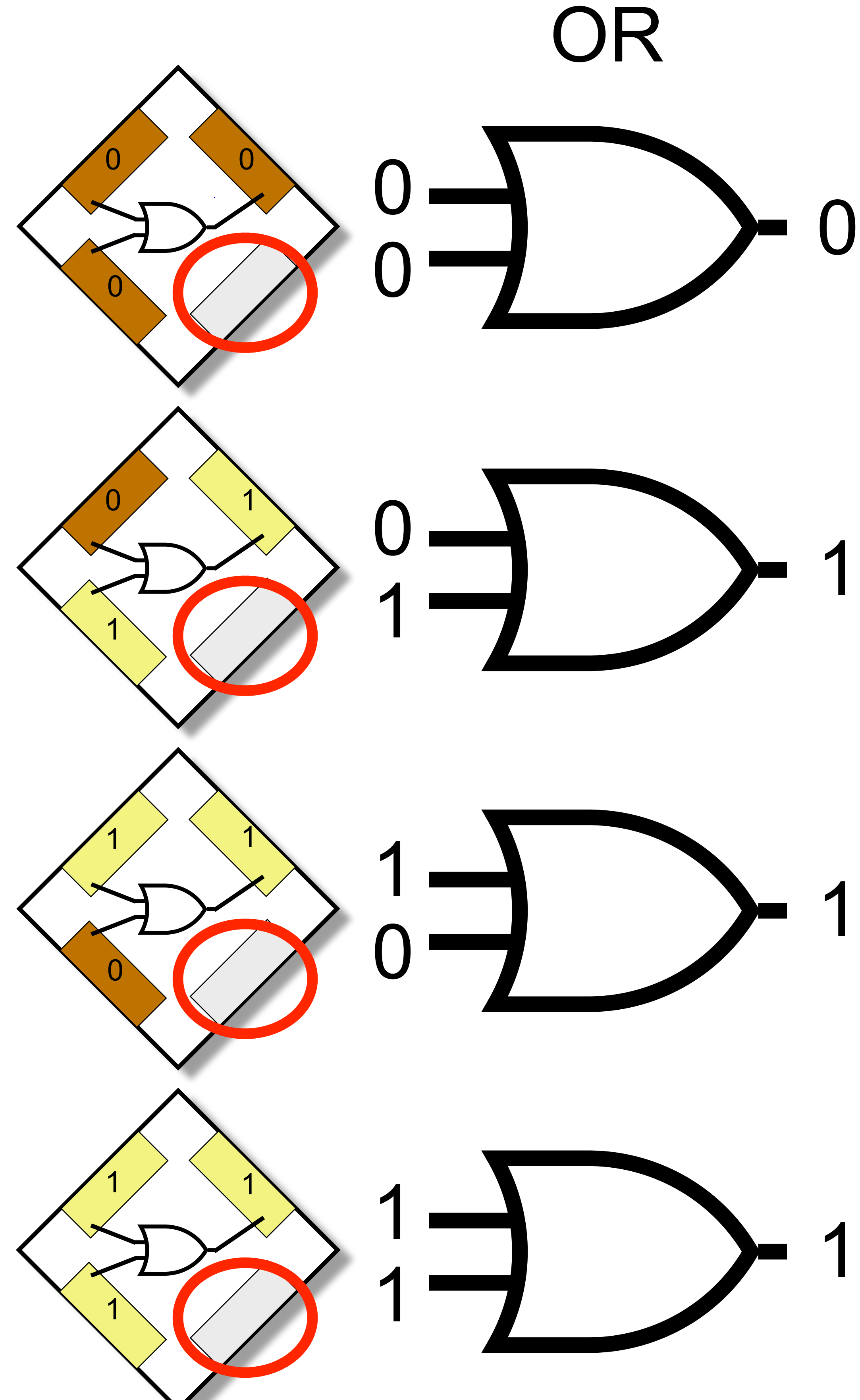




Smart self-assembly

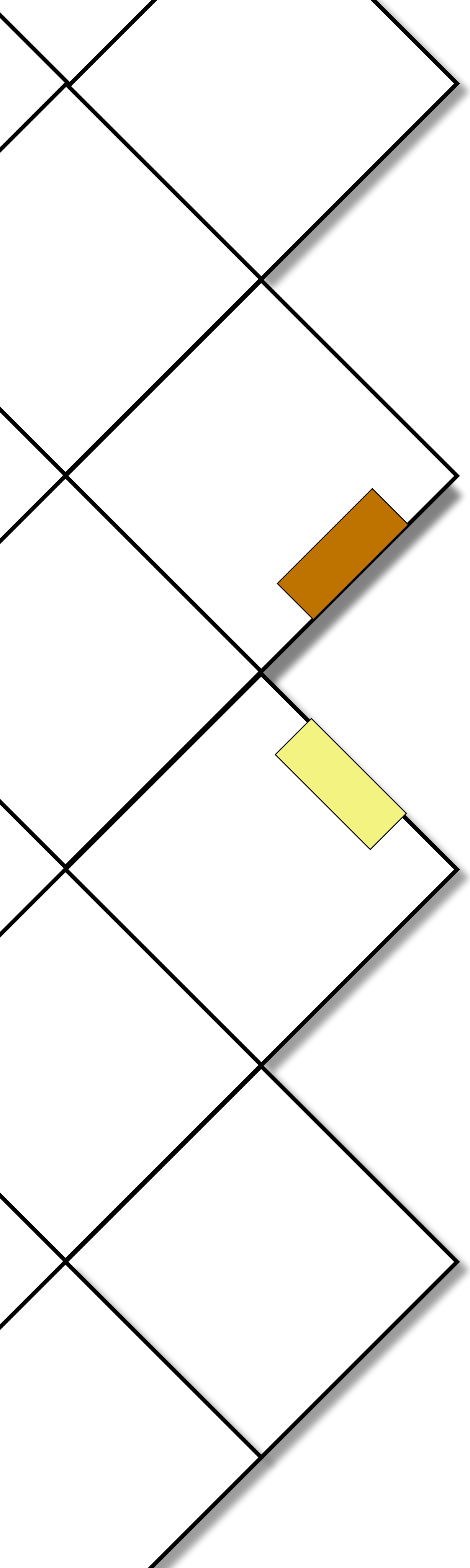
logic gates: simple, yet powerful

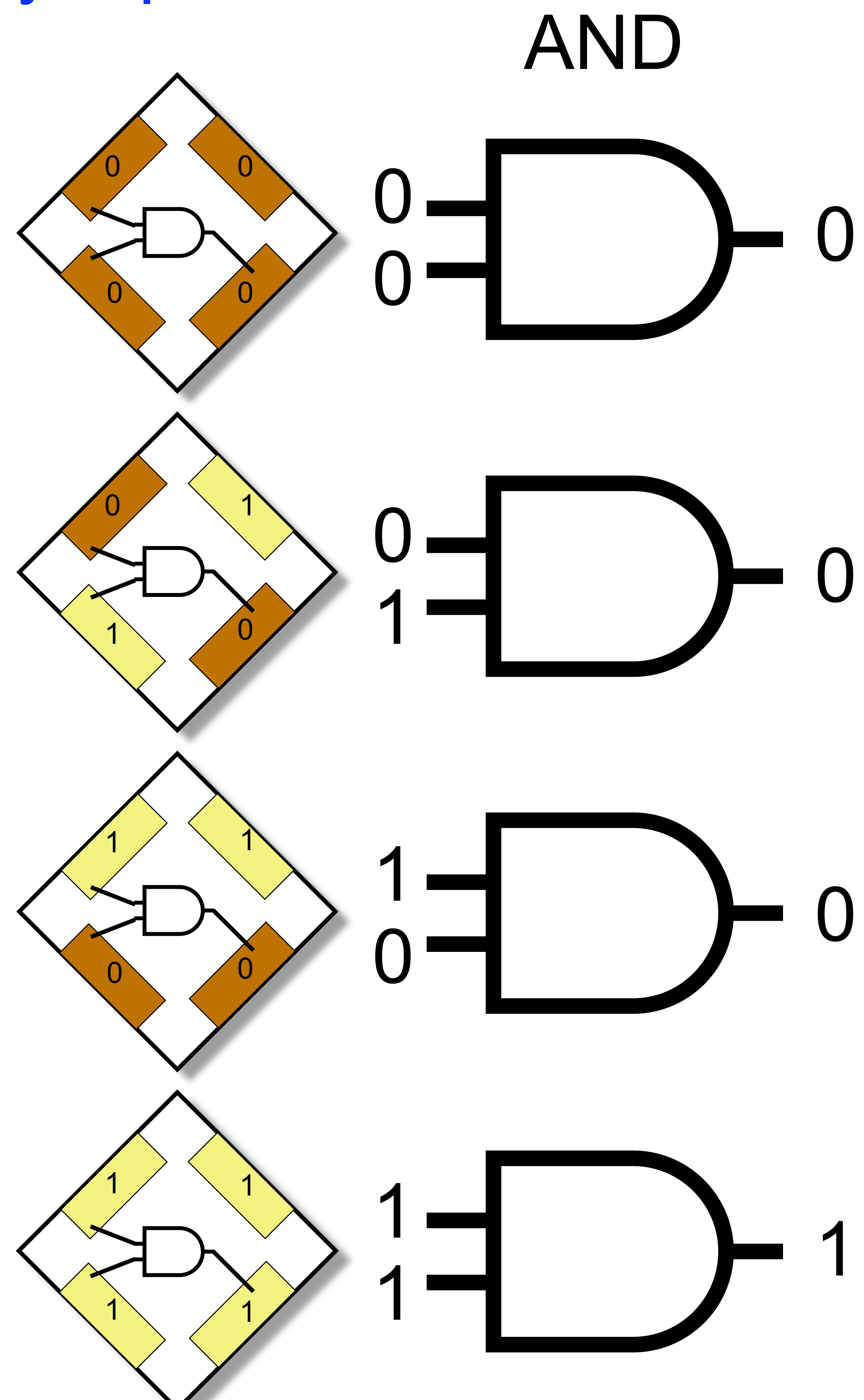




Smart self-assembly

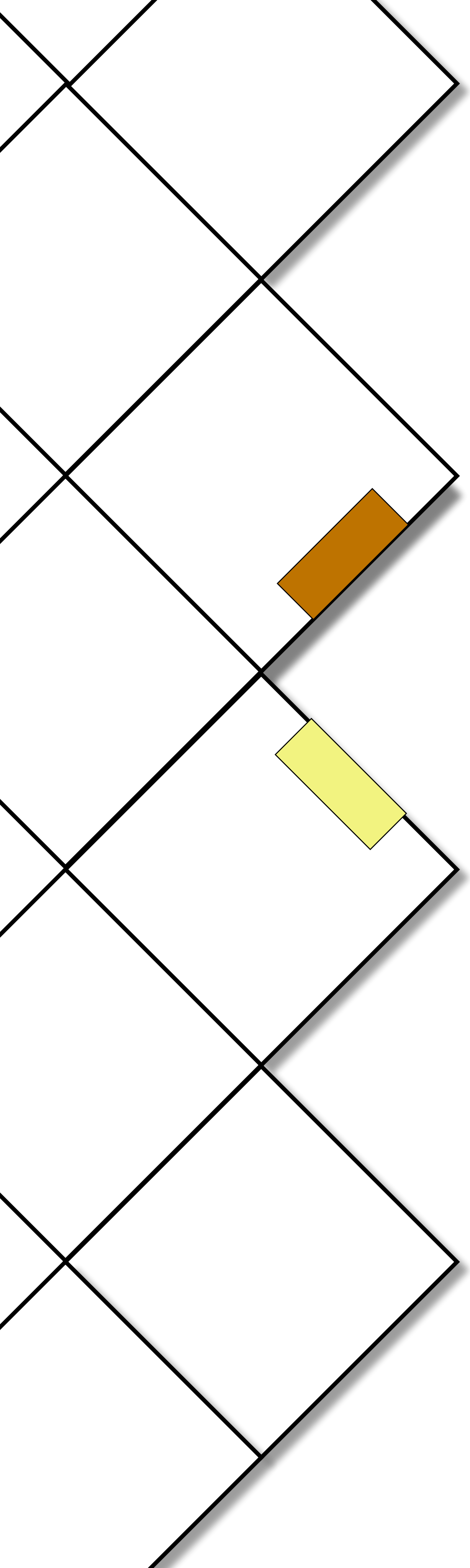
logic gates: simple, yet powerful

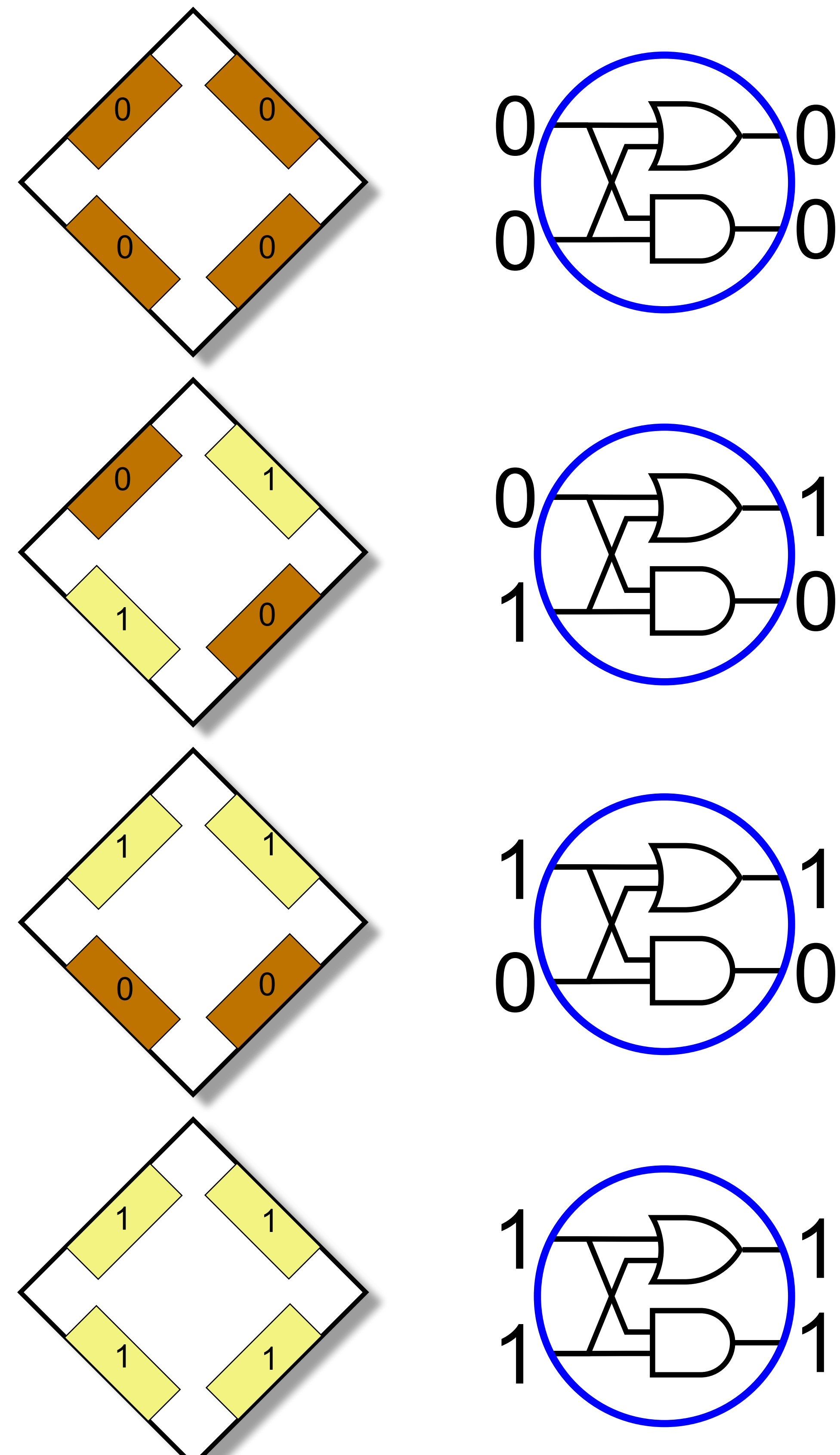




Smart self-assembly

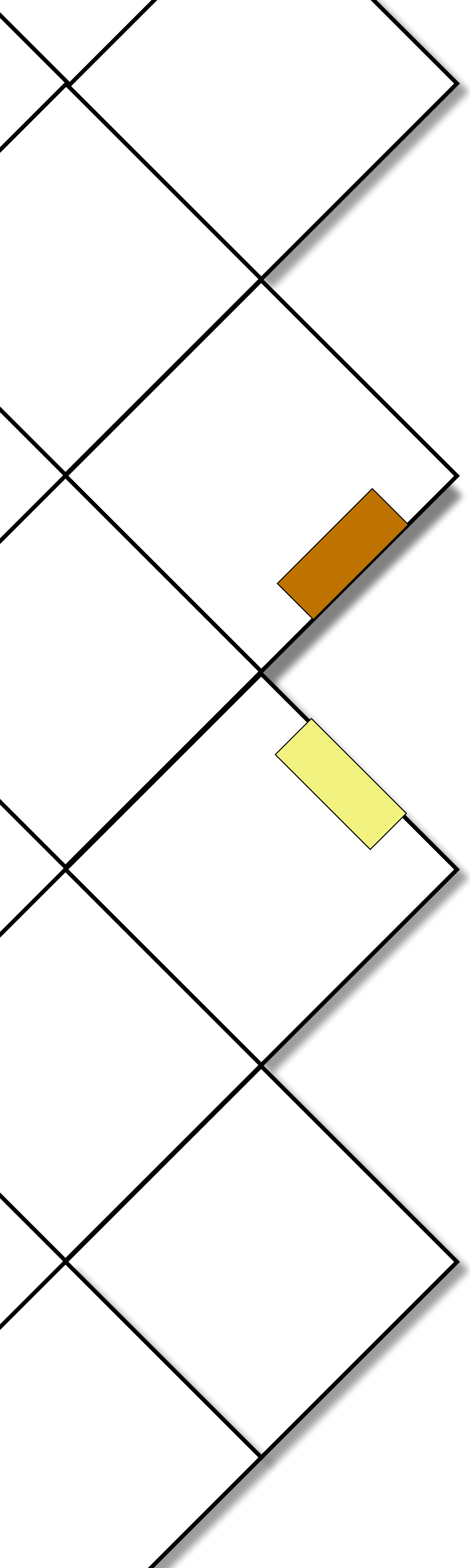
logic gates: simple, yet powerful

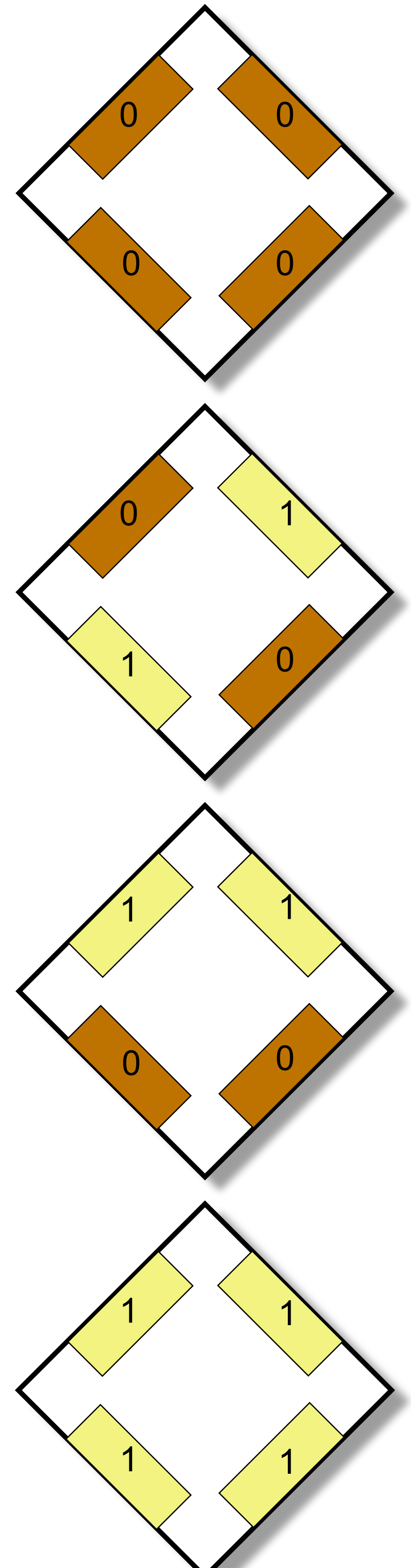


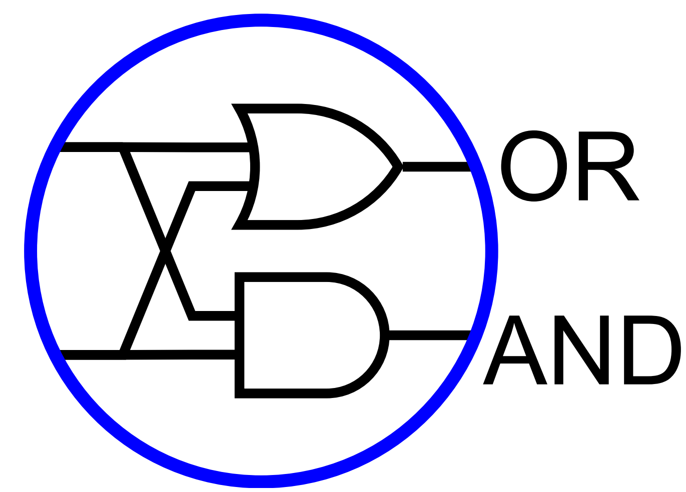


Smart self-assembly

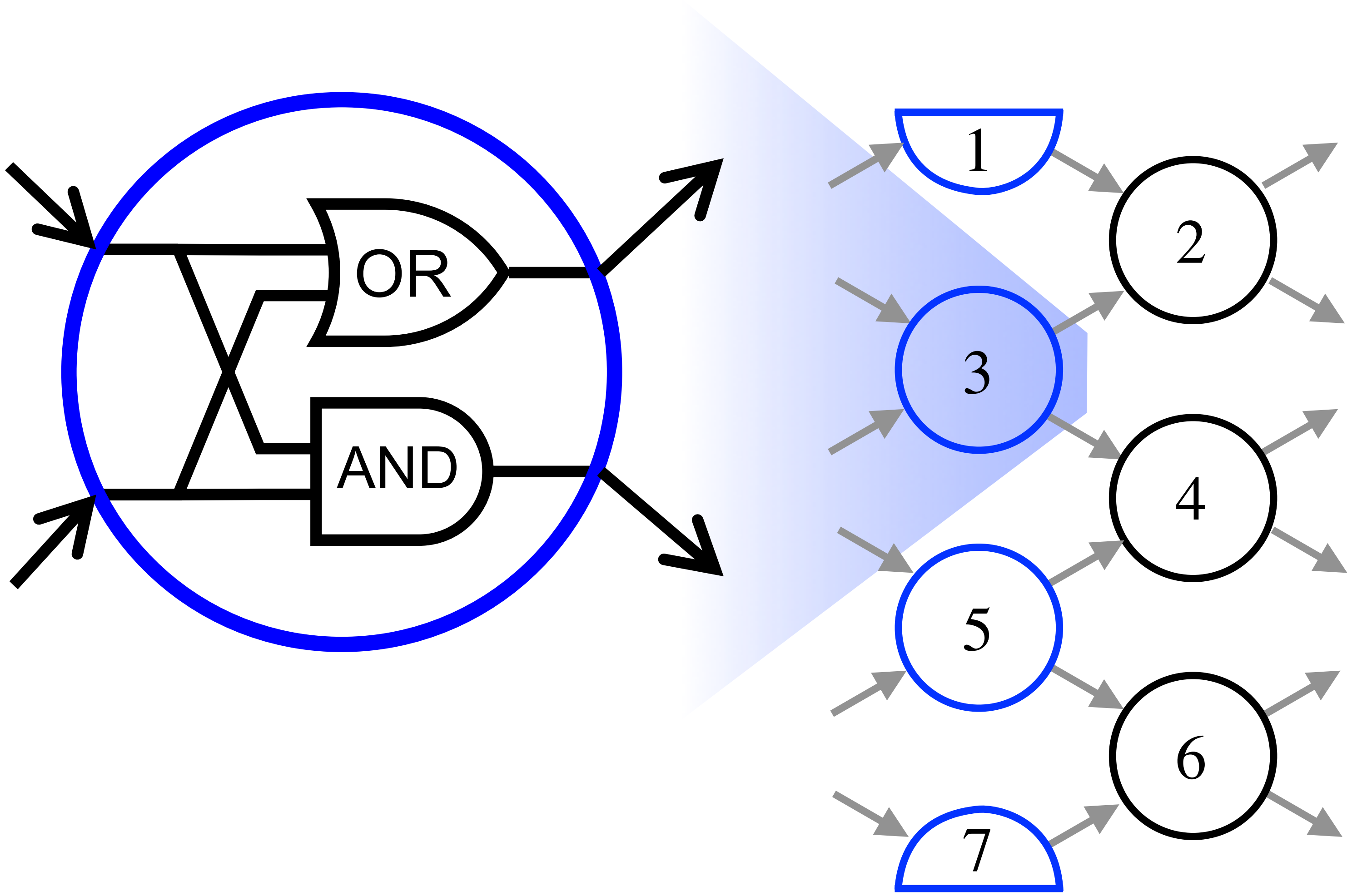
logic gates: simple, yet powerful



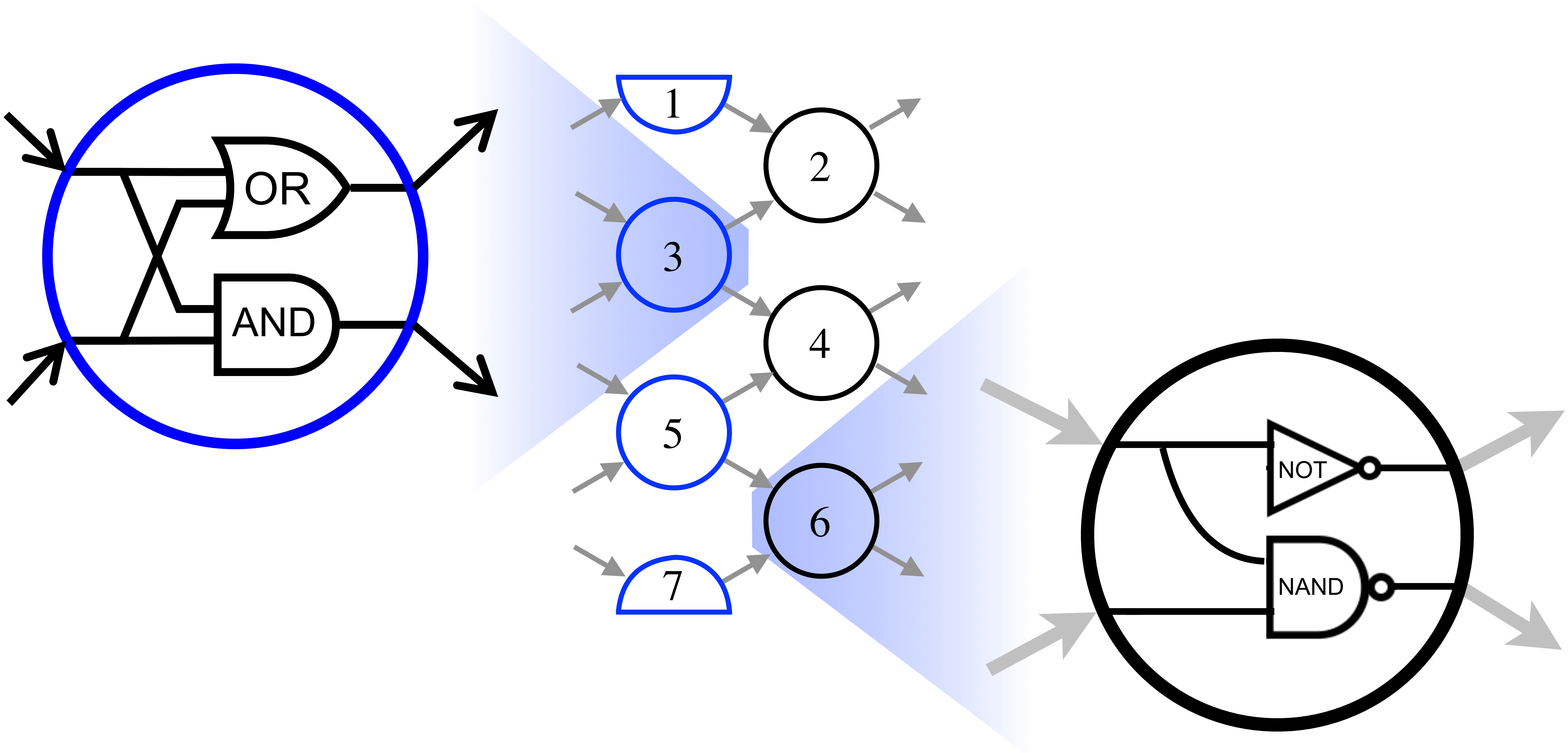




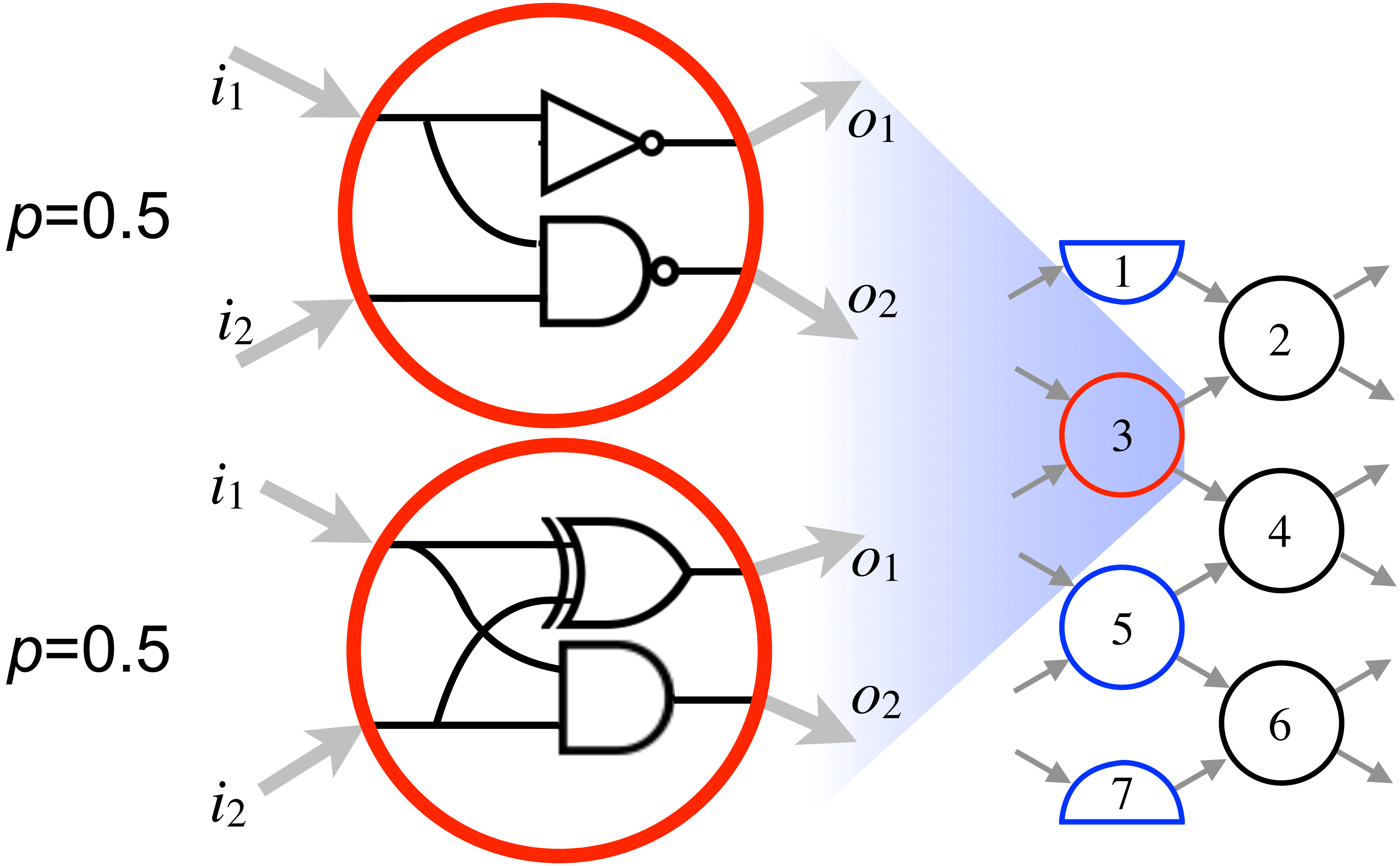
A local Boolean circuit model



A local Boolean circuit model

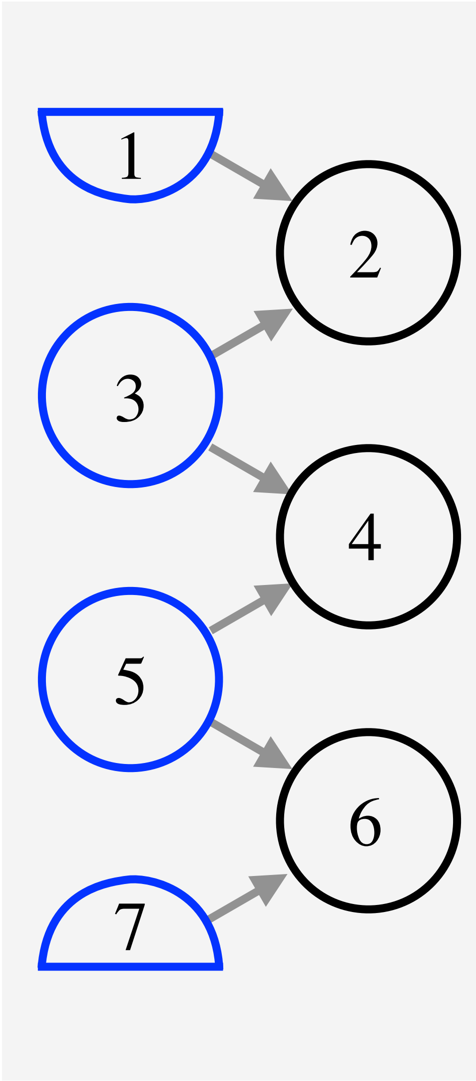


A local circuit model: randomised gates



A local Boolean circuit model

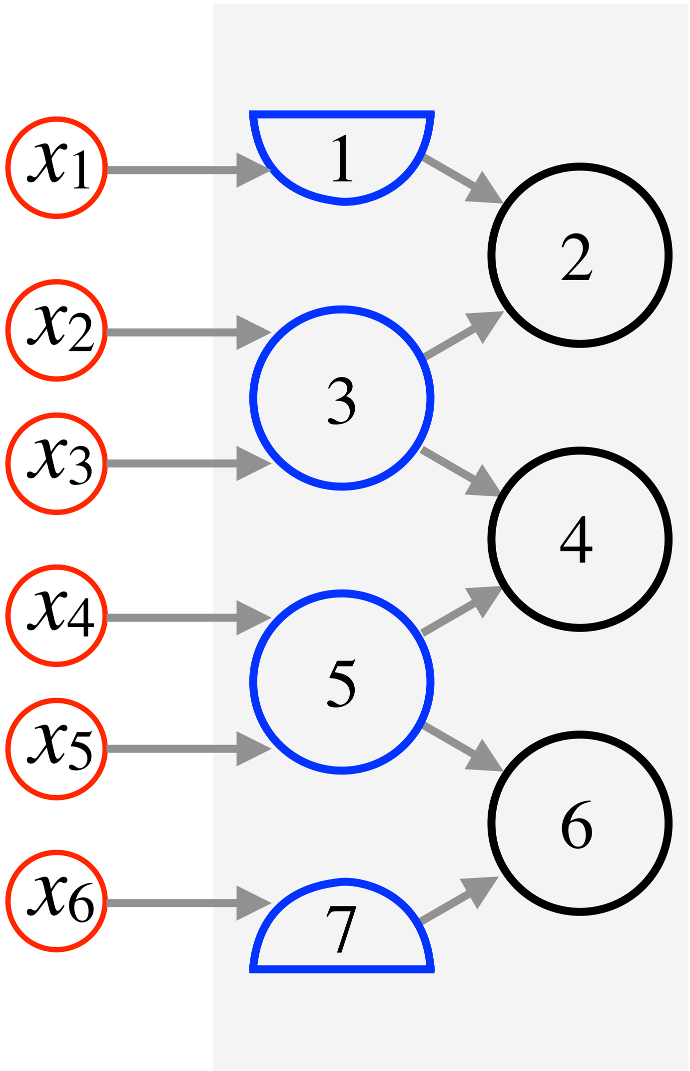
Programmer
specifies a layer



A local Boolean circuit model

Programmer specifies a layer

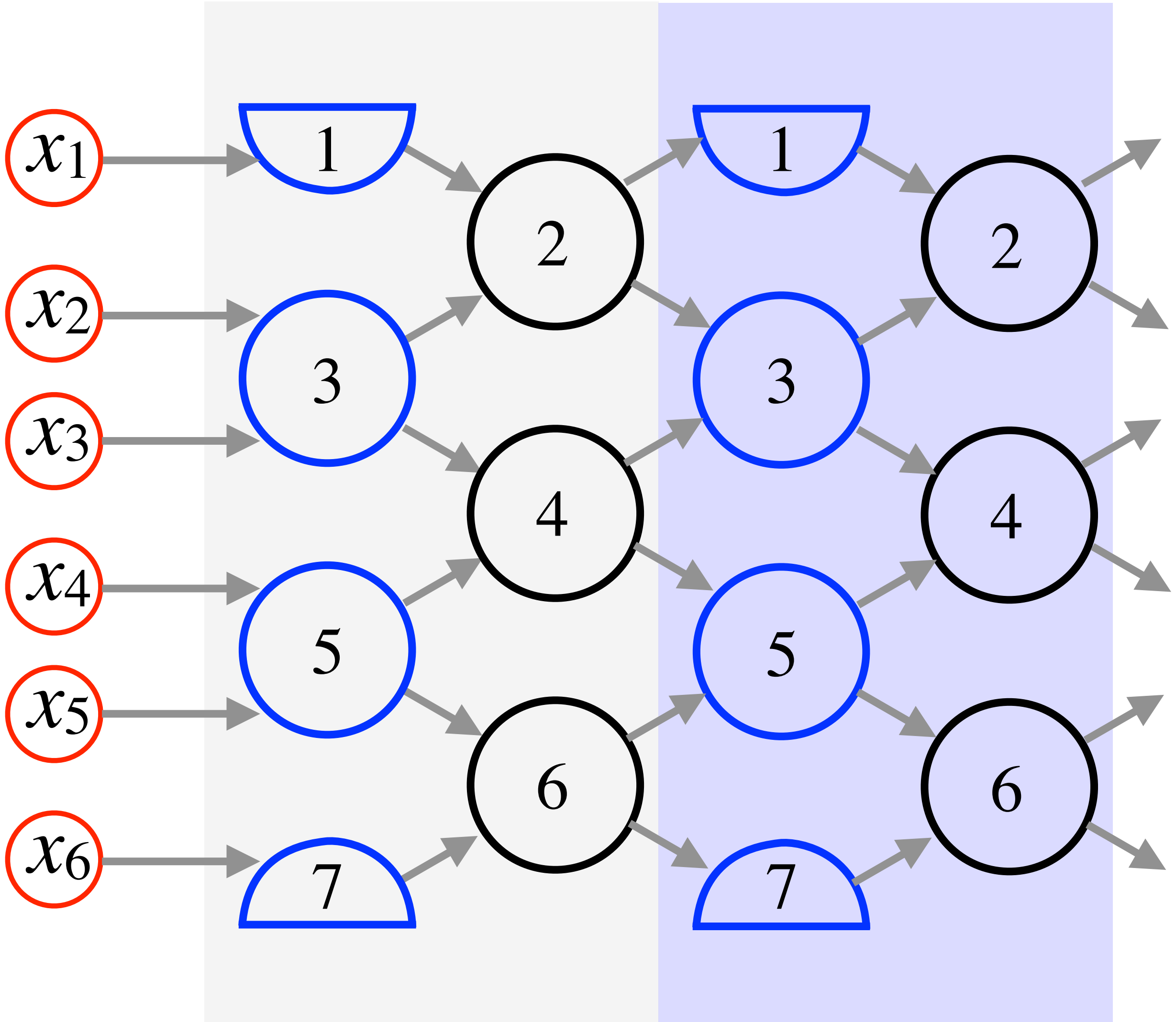
User gives n input bits $x_k \in \{0,1\}$



A local Boolean circuit model

Programmer specifies a layer

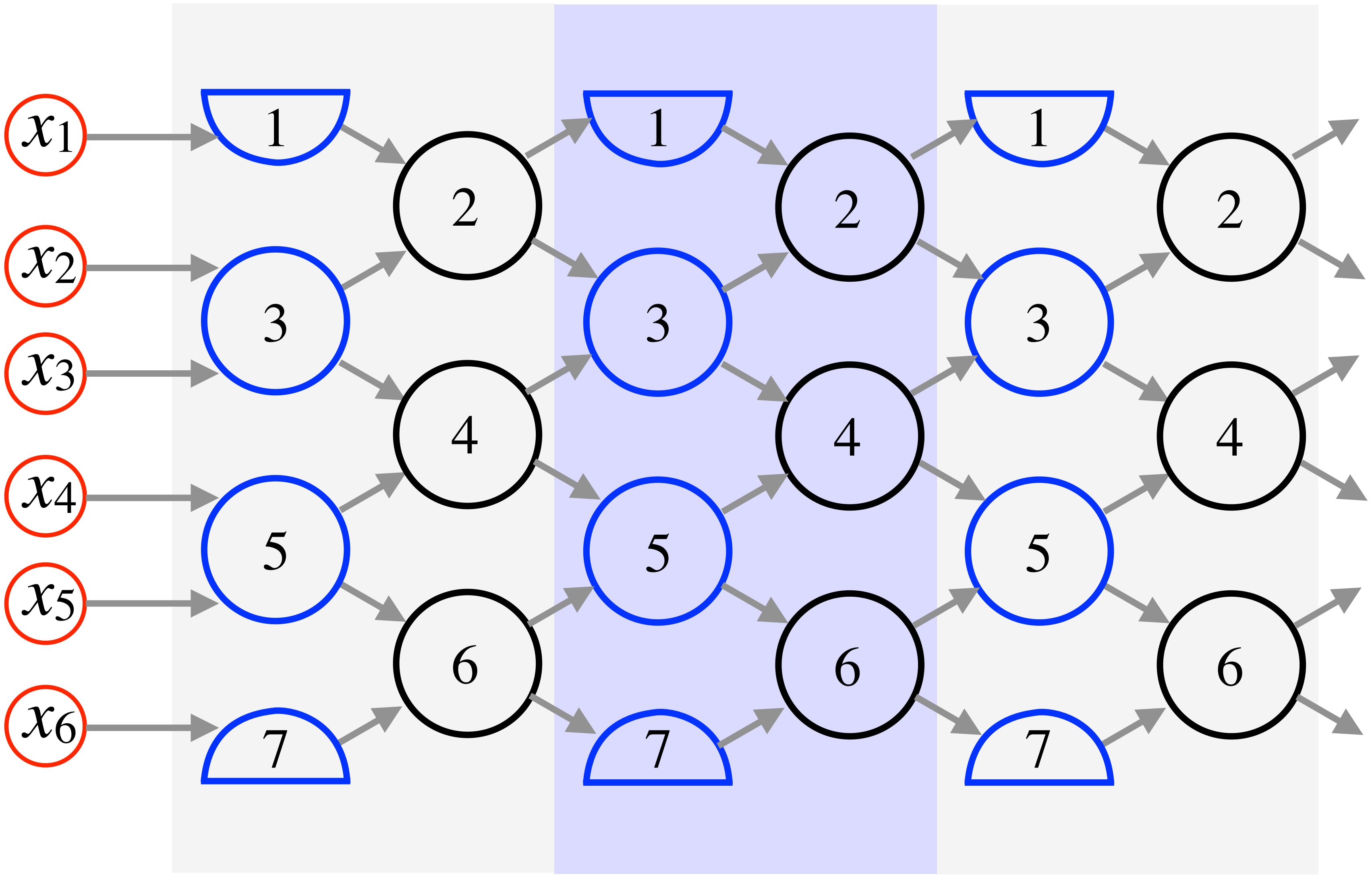
User gives n input bits $x_k \in \{0,1\}$



A local Boolean circuit model

Programmer specifies a layer

User gives *n* input bits $x_k \in \{0,1\}$

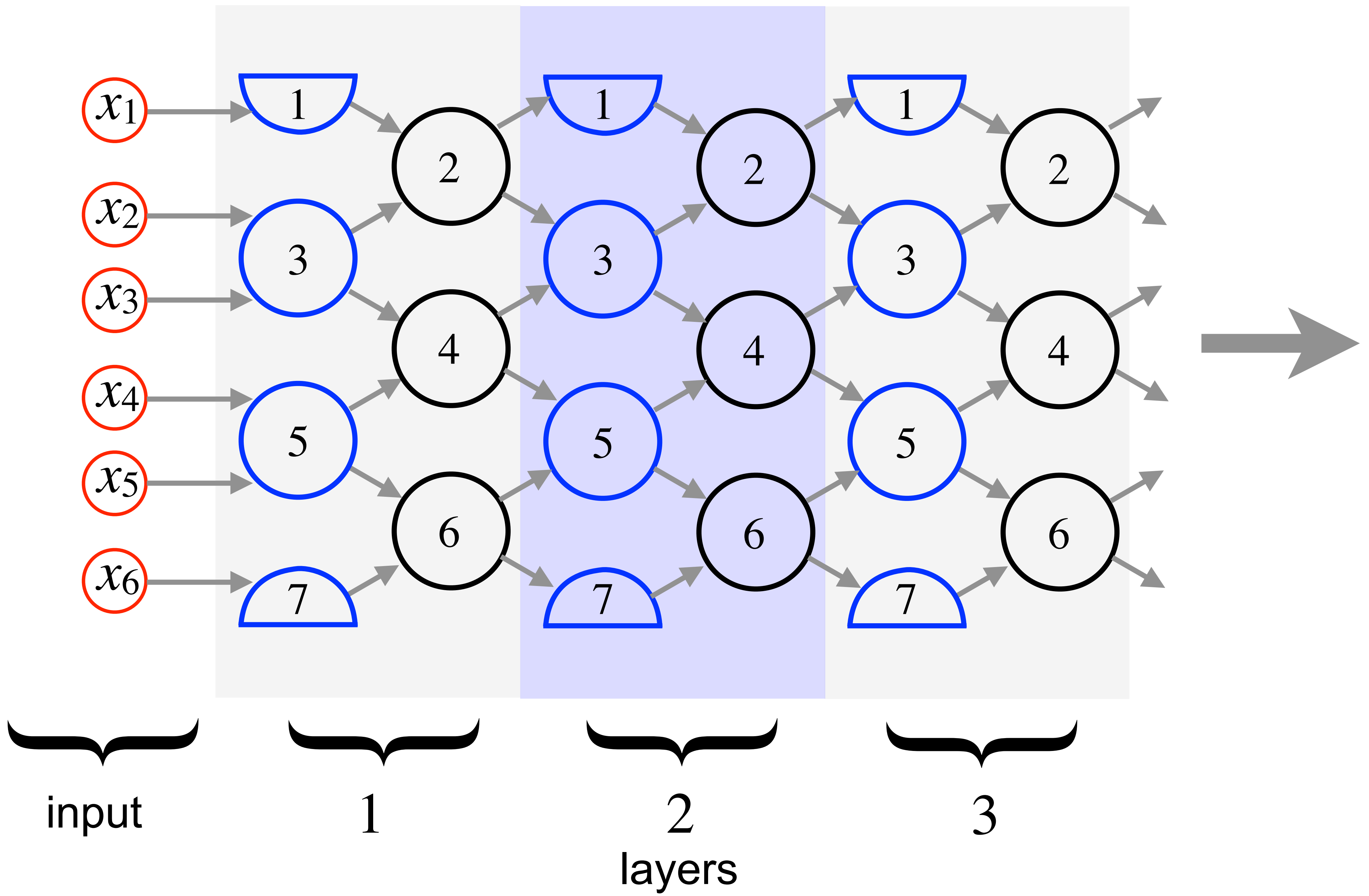


A local Boolean circuit model

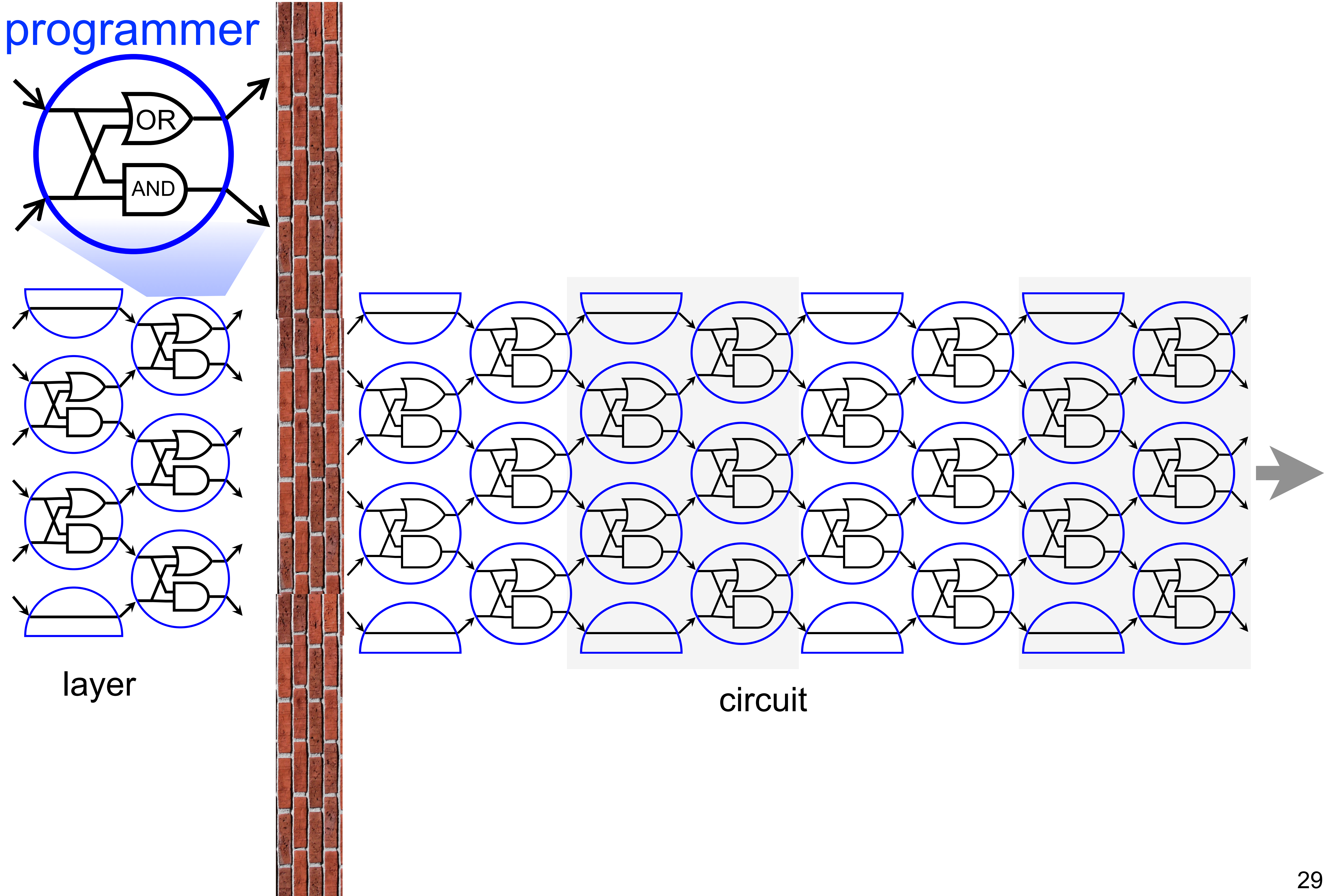
Programmer specifies a layer

User gives *n* input bits $x_k \in \{0,1\}$

Computation flows from input gates to layer 1, layer 2, layer 3 …

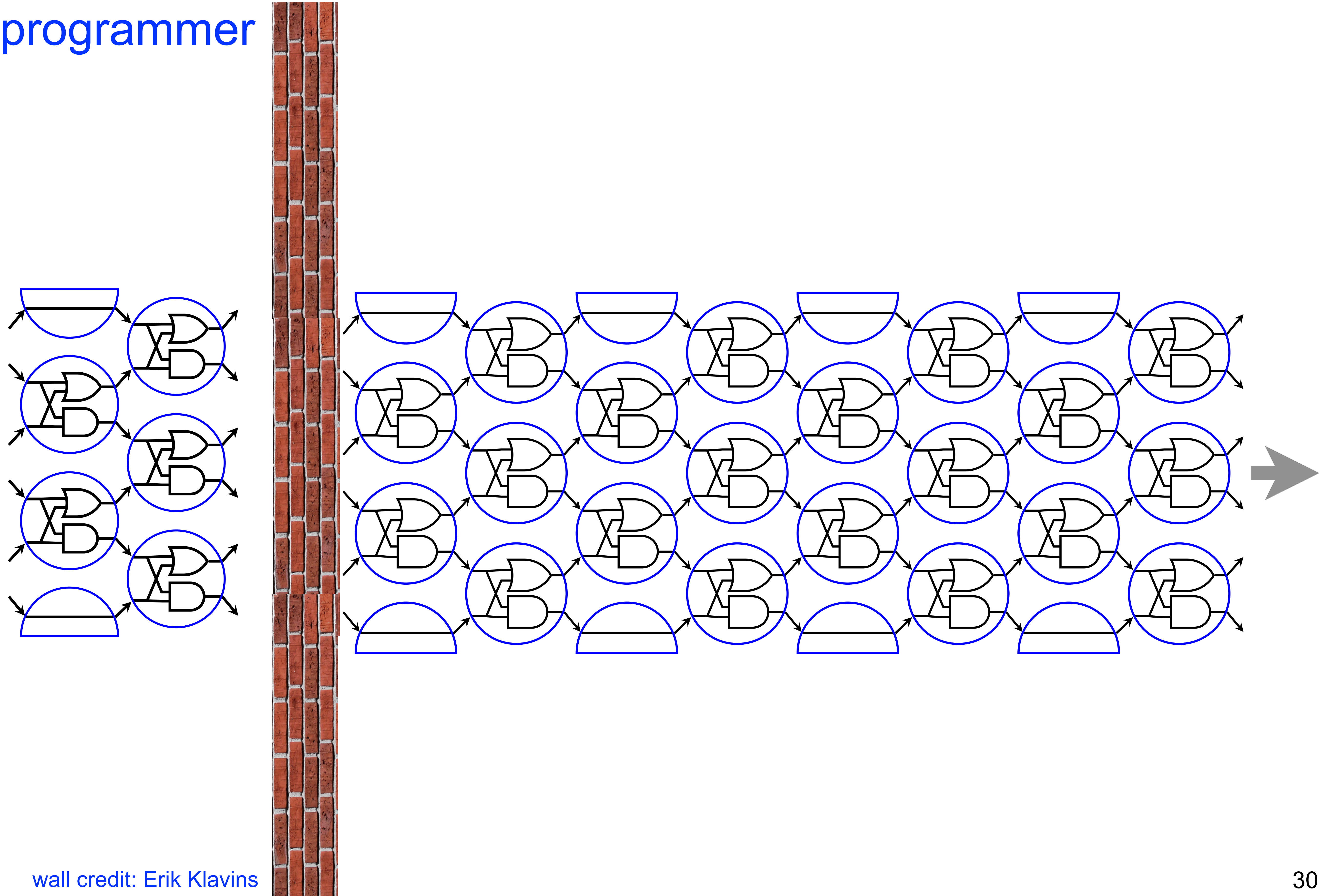


Example circuit



Example circuit

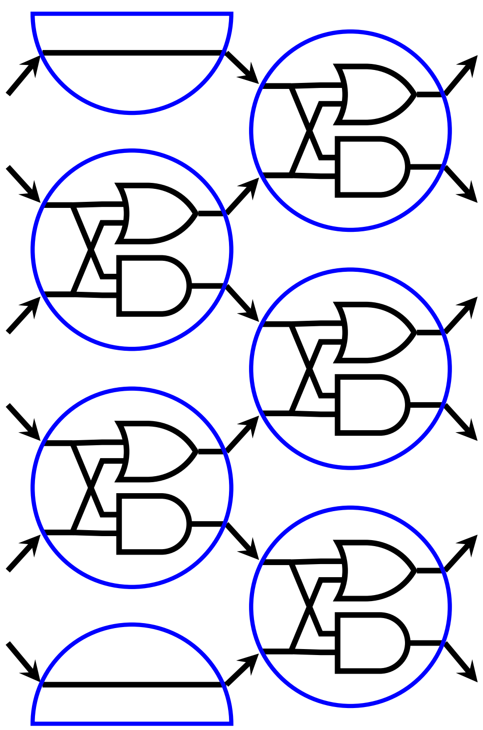
programmer



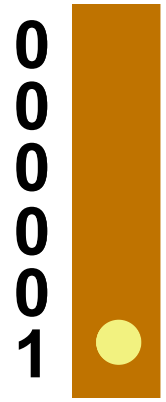
wall credit: Erik Klavins

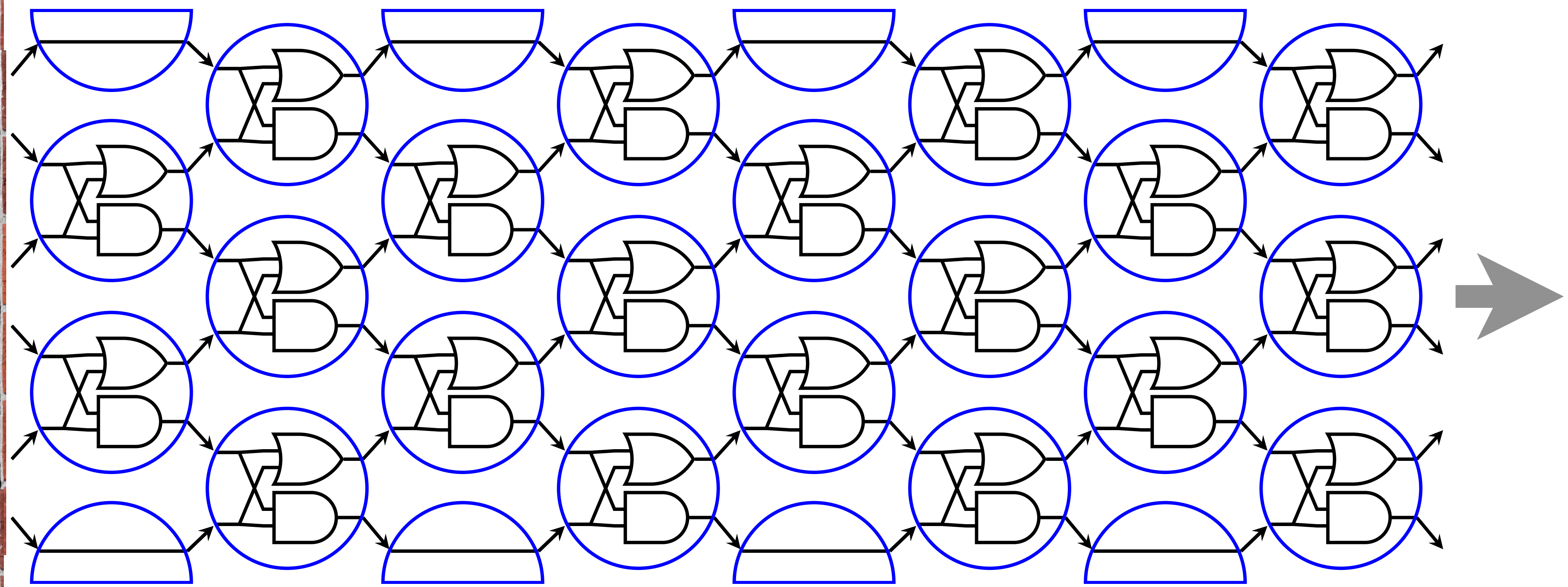
Example circuit

programmer



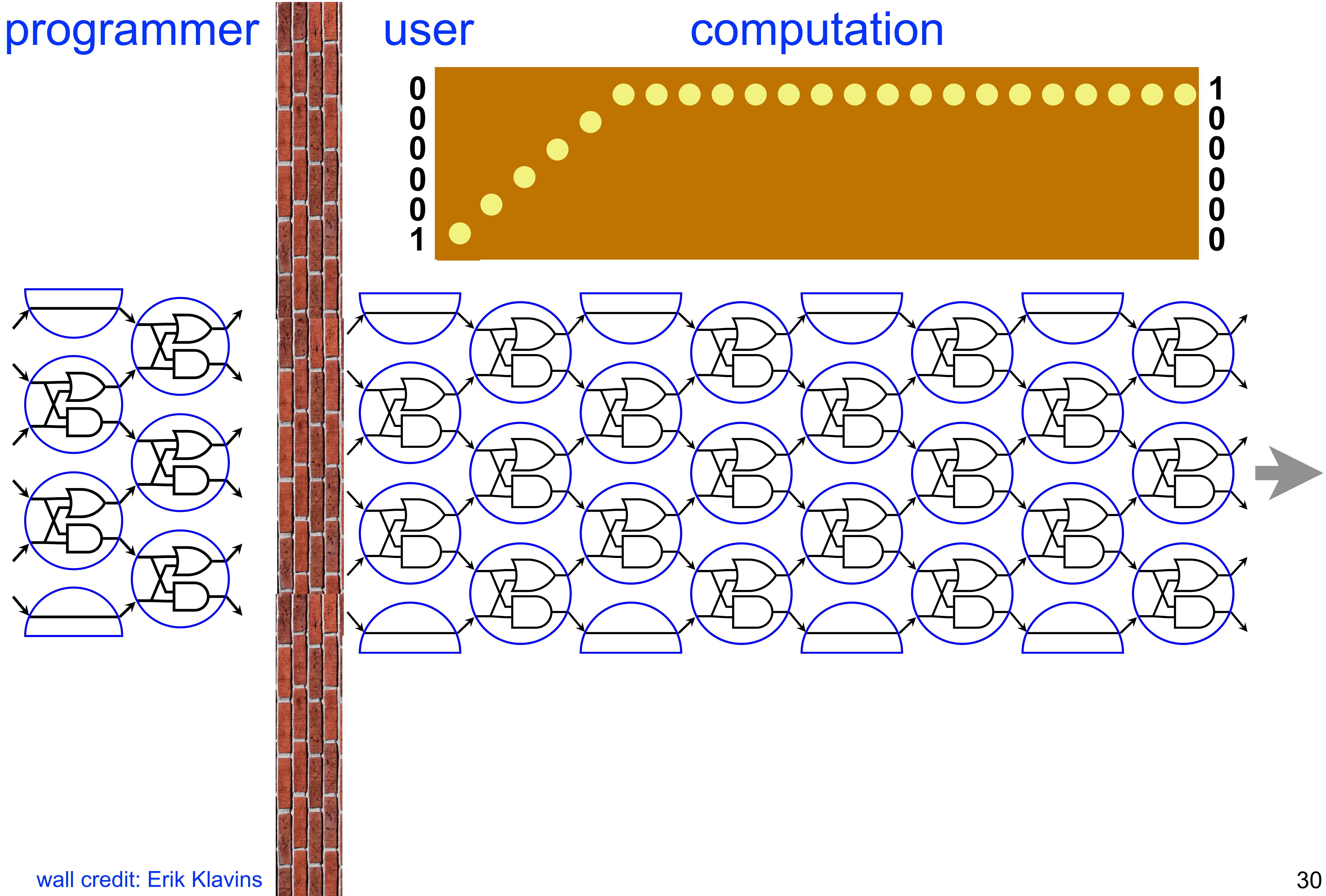
user





wall credit: Erik Klavins

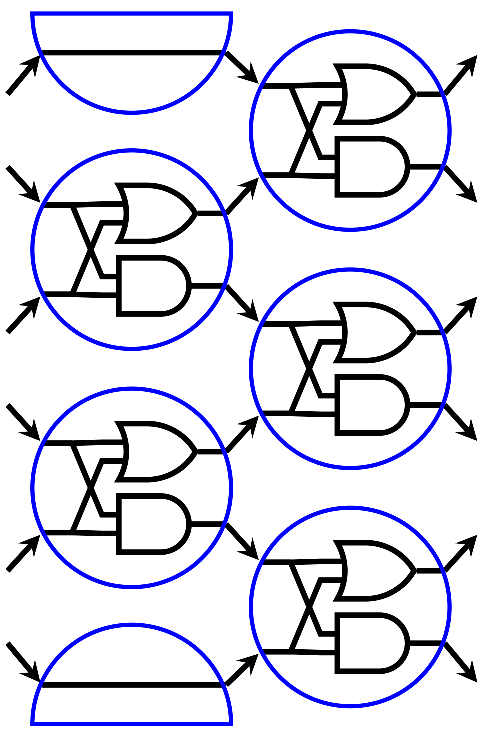
Example circuit



wall credit: Erik Klavins

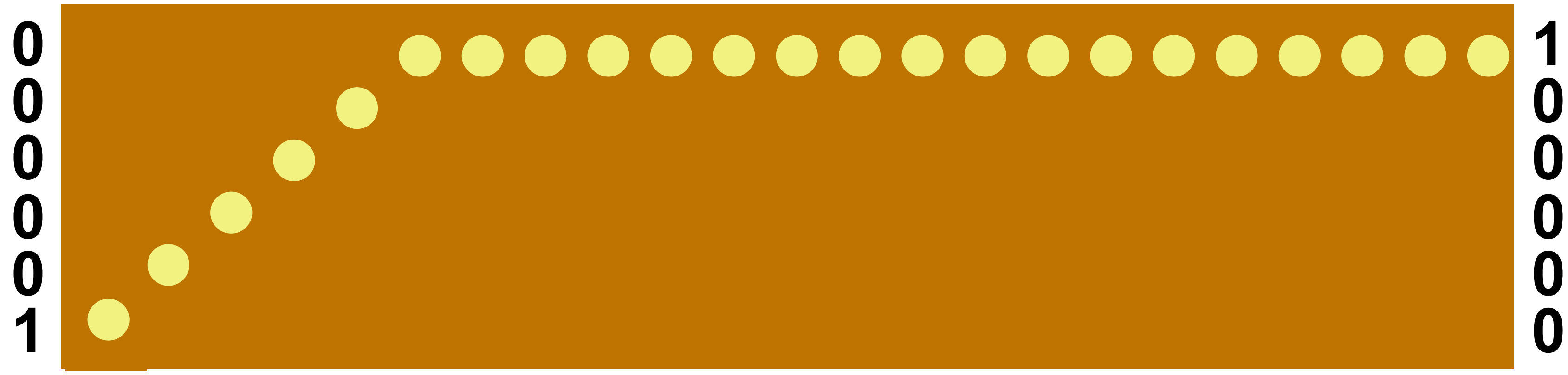
Example circuit

programmer



user

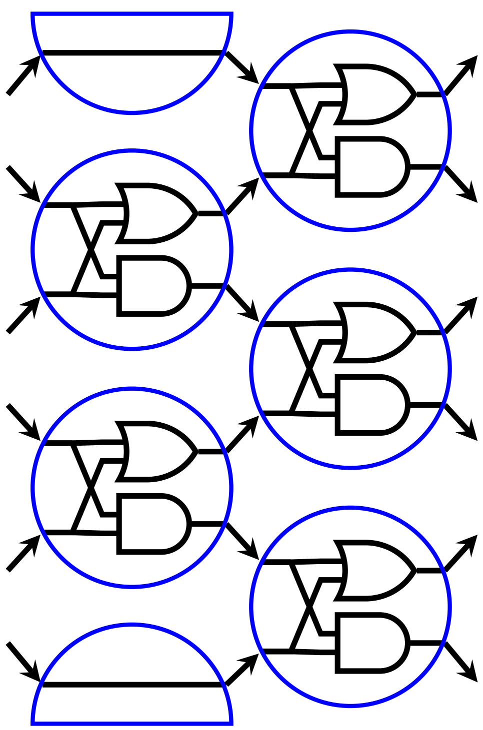
computation



wall credit: Erik Klavins

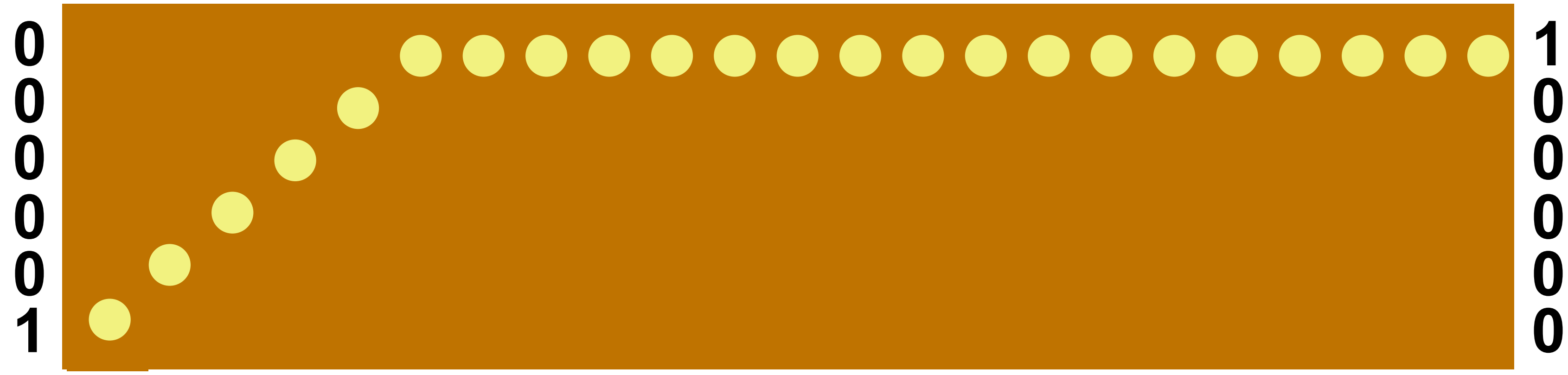
Example circuit: “SORTING”

programmer



user

computation



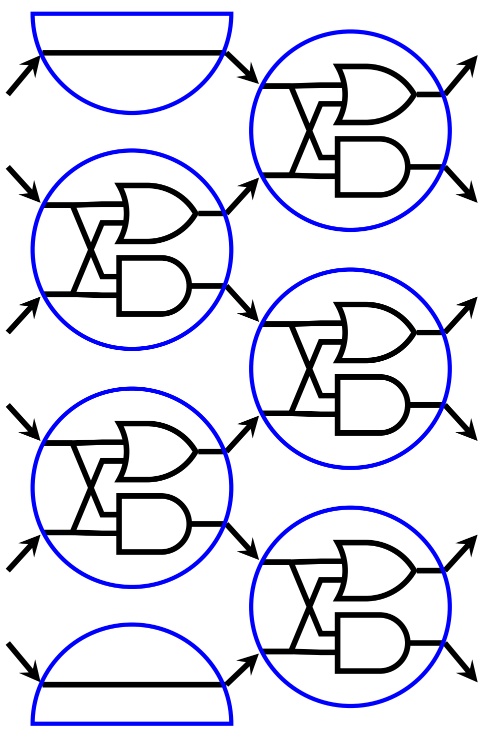
wall credit: Erik Klavins

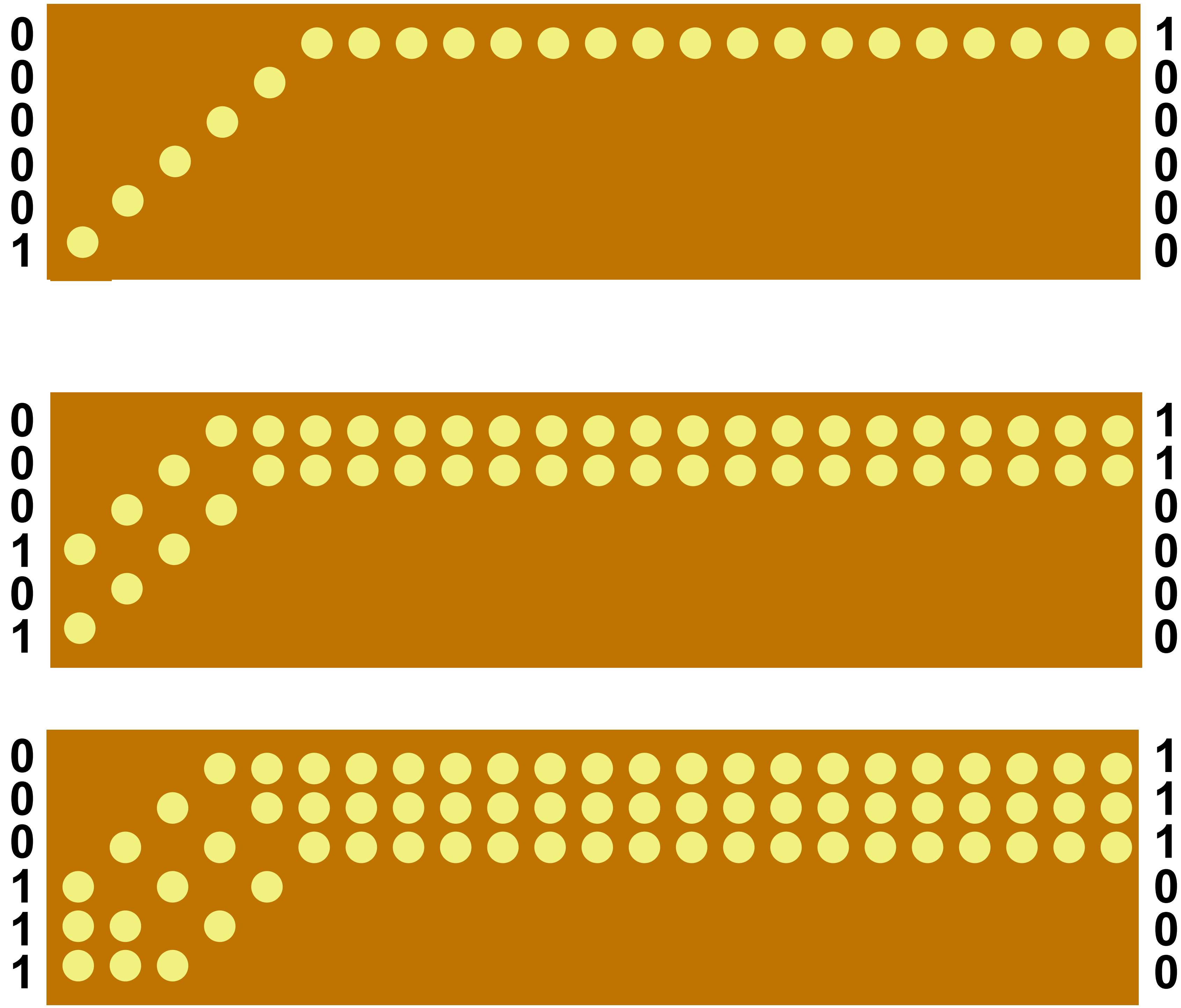
Example circuit: “SORTING”

programmer

user

computation





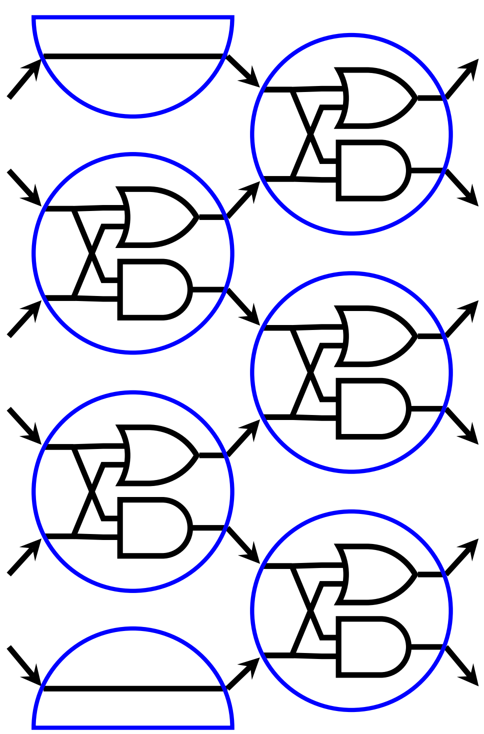
wall credit: Erik Klavins

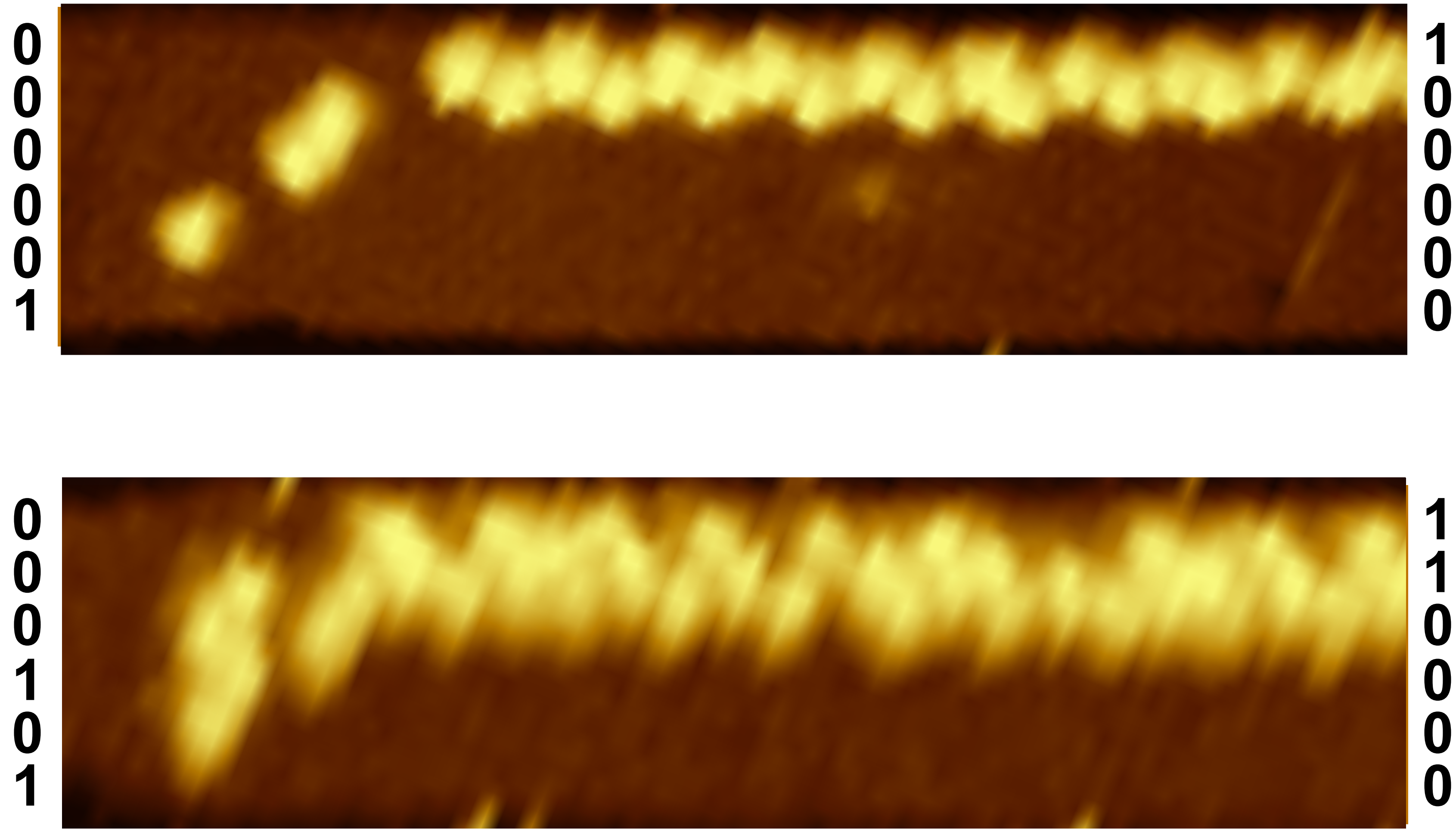
Example circuit: “SORTING”

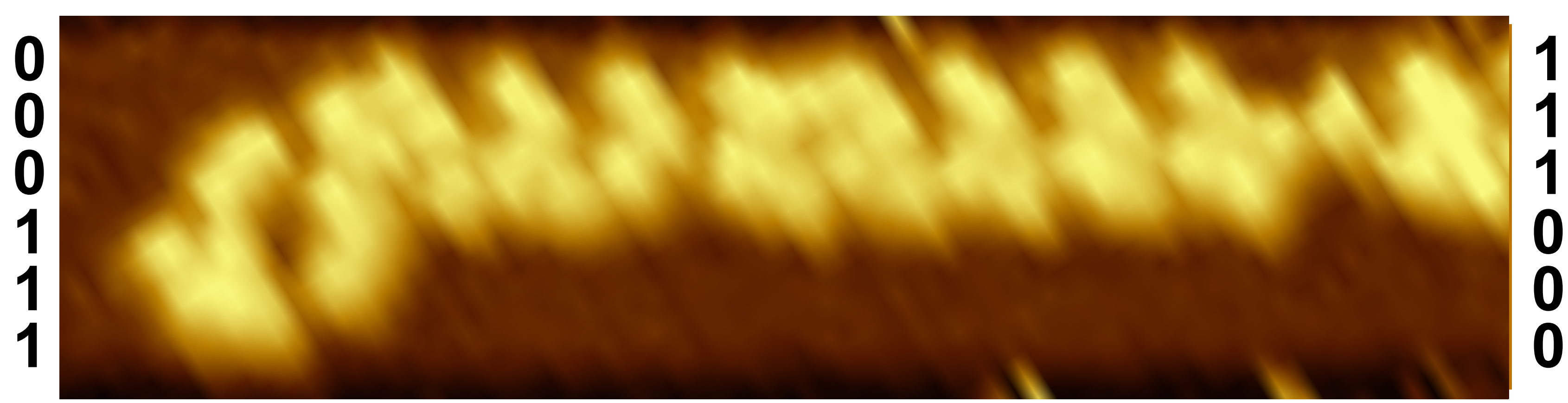
programmer

user

computation

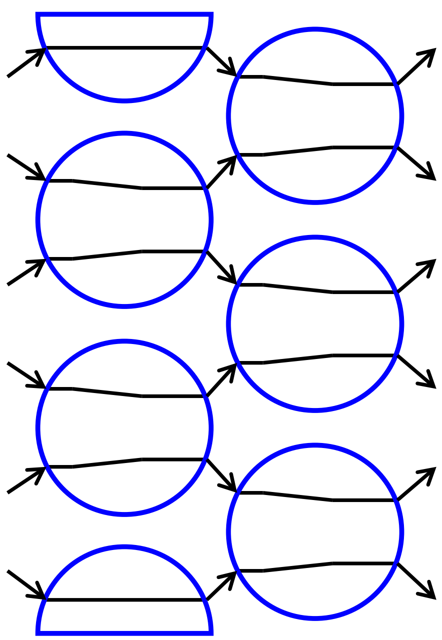


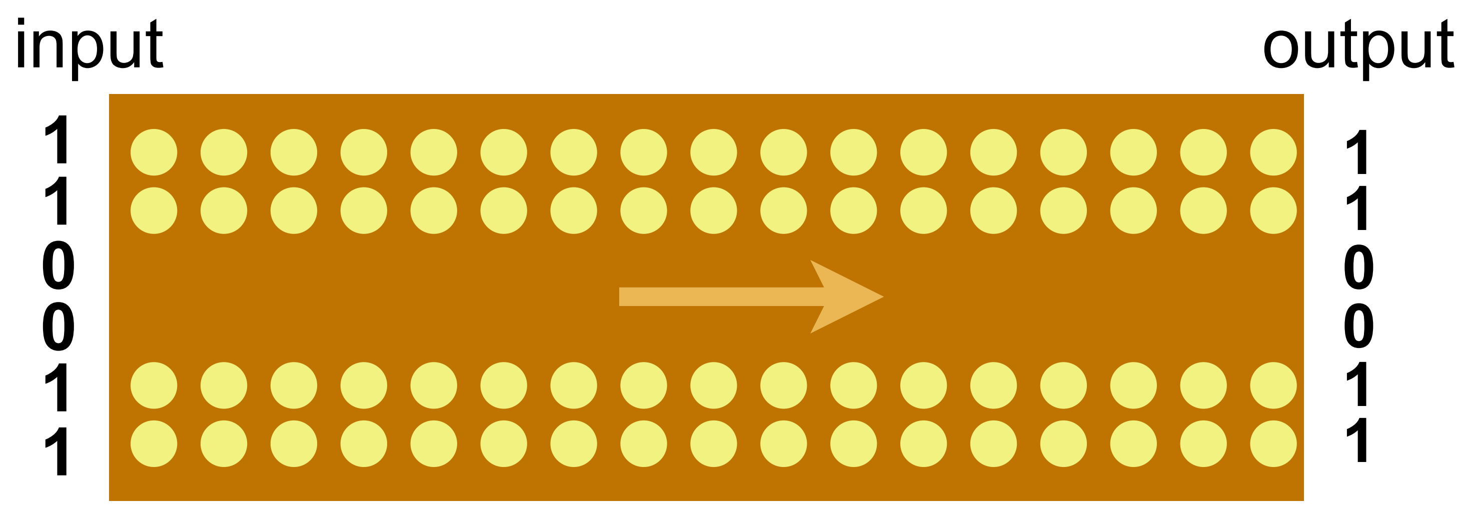




wall credit: Erik Klavins

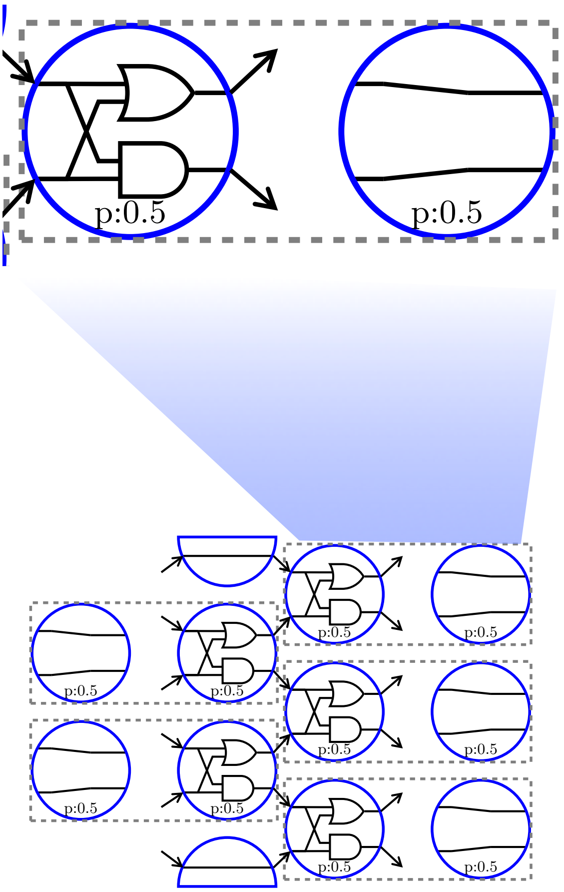
Example circuit: COPY bits to the right





Example circuit: LAZYSORTING

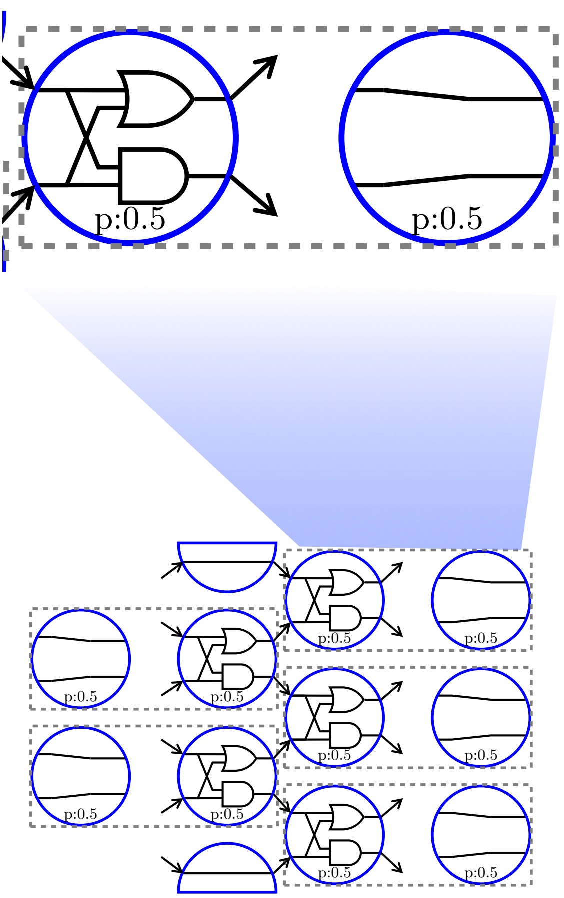
programmer



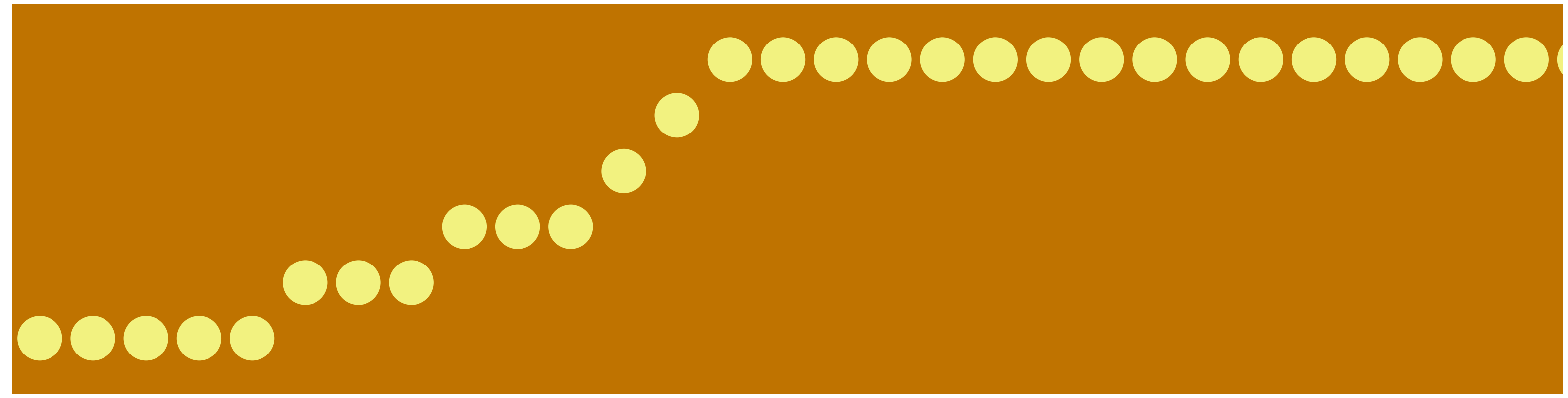
wall credit: Erik Klavins

Example circuit: LAZYSORTING

programmer | user | computation

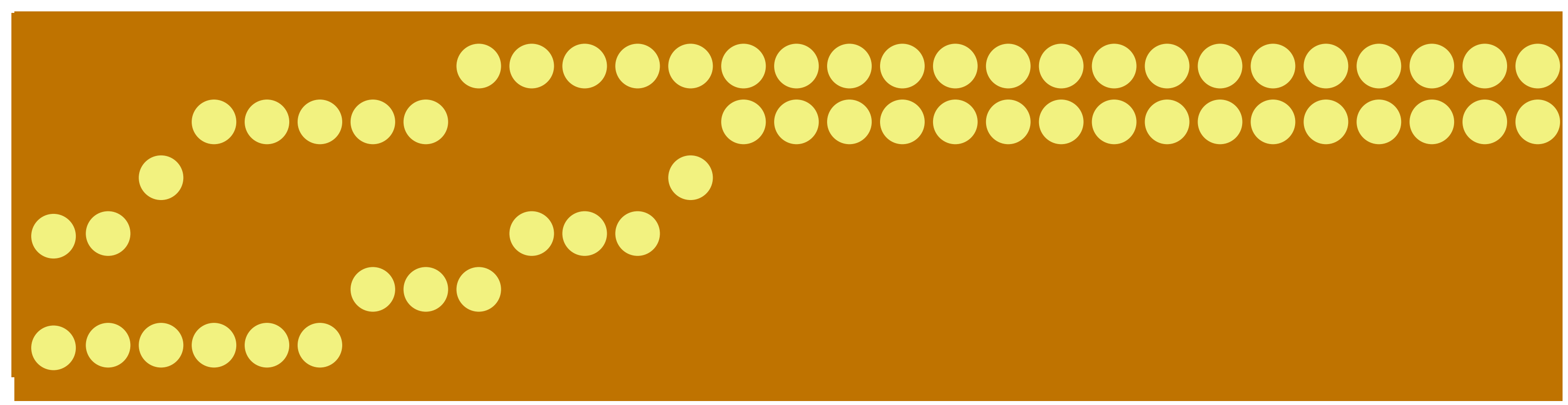


0 0 0 0 0 1



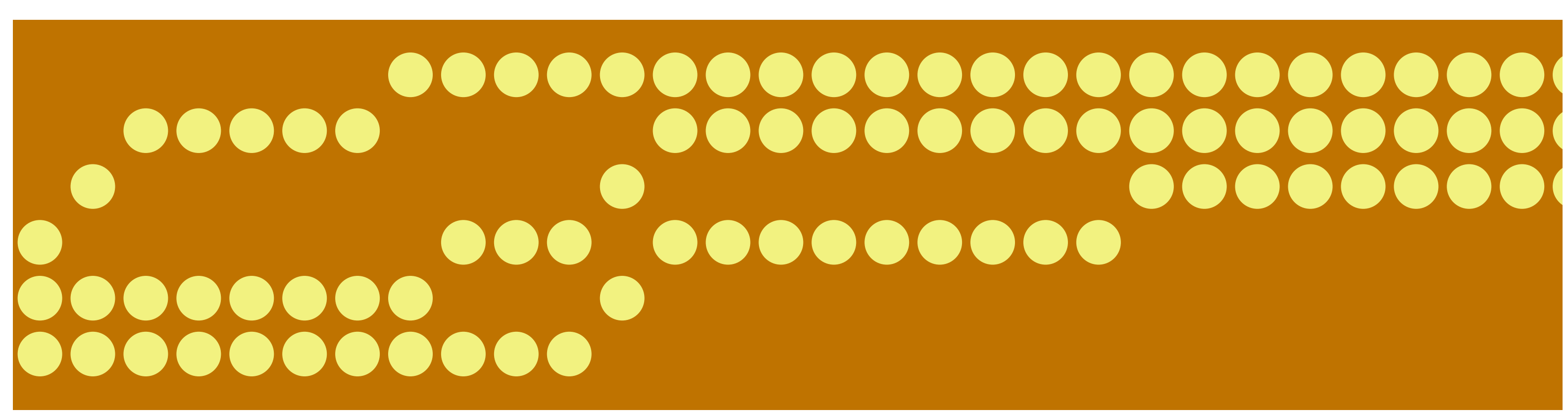
1 0 0 0 0 0

0 0 0 1 0 1



1 1 0 0 0 0

0 0 0 1 1 1

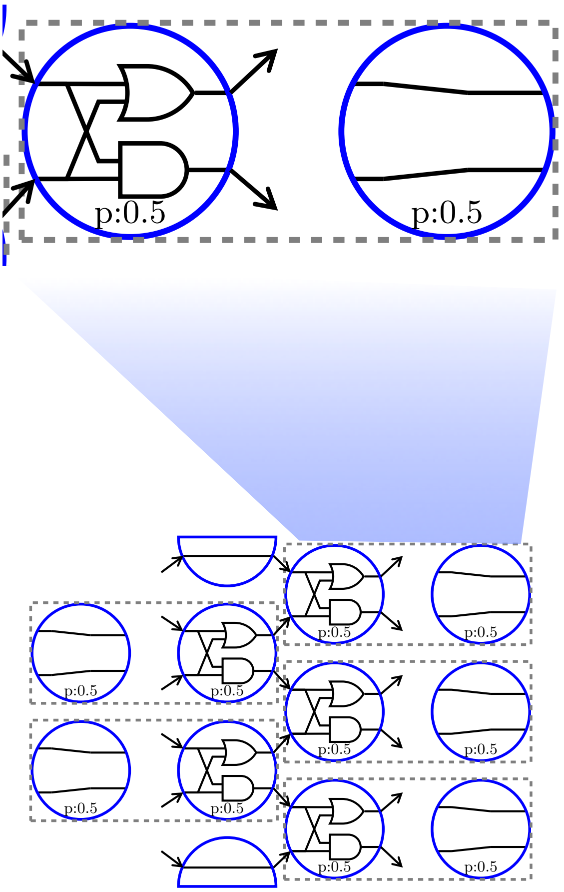


1 1 1 0 0 0

wall credit: Erik Klavins

Example circuit: LAZYSORTING

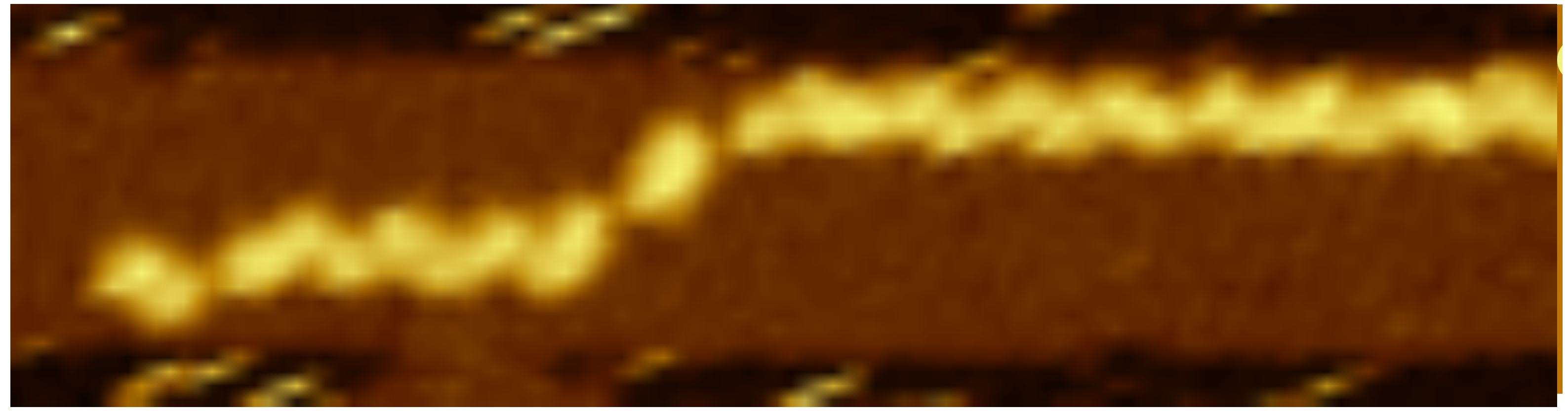
programmer



user

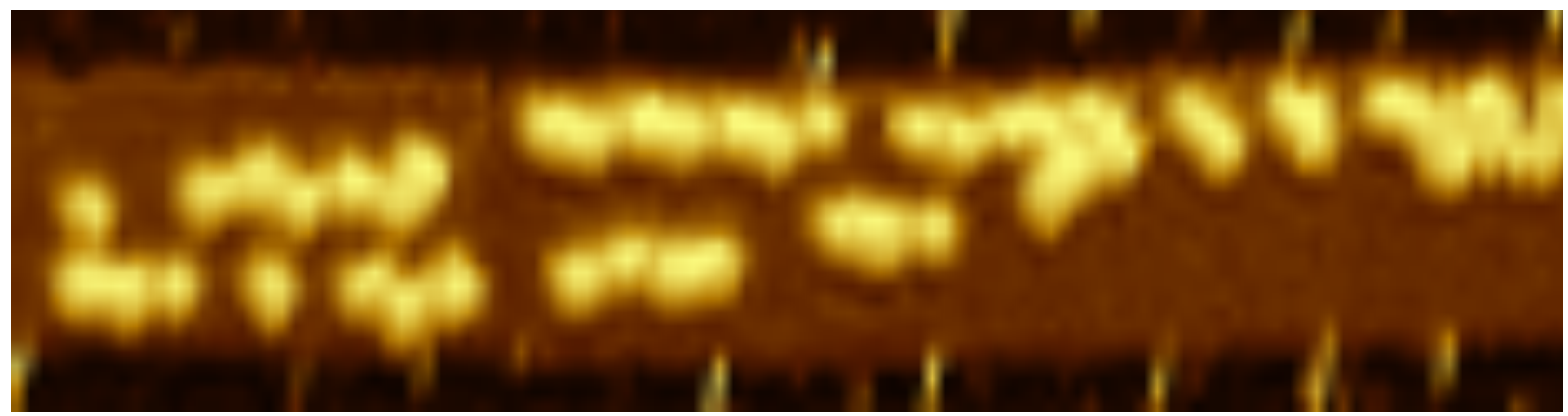
computation

0 0 0 0 0 1



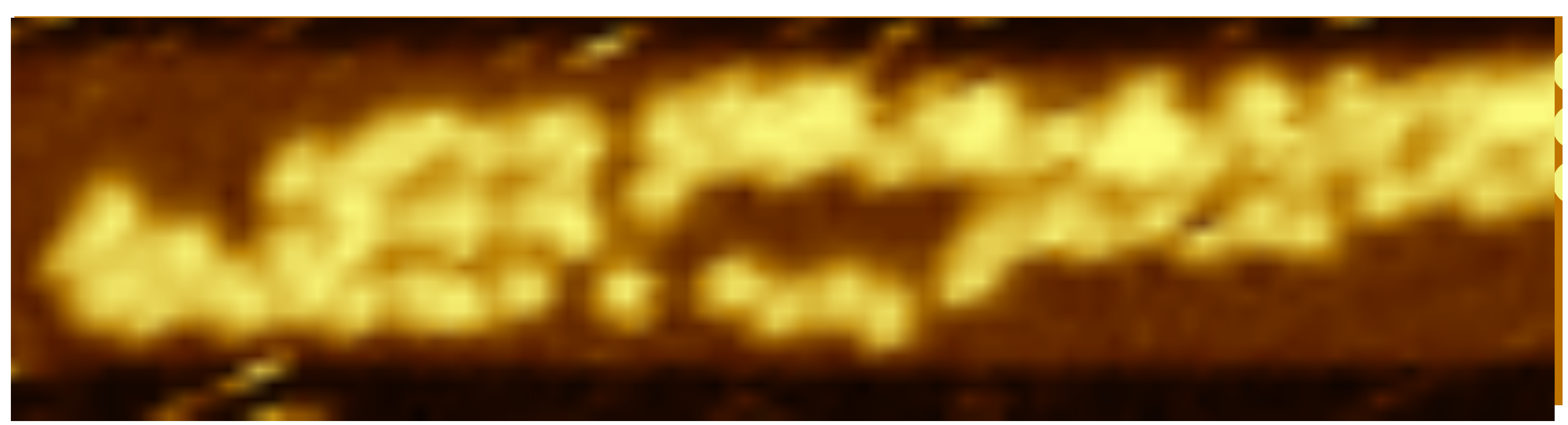
1 0 0 0 0 0

0 0 0 1 0 1



1 1 0 0 0 0

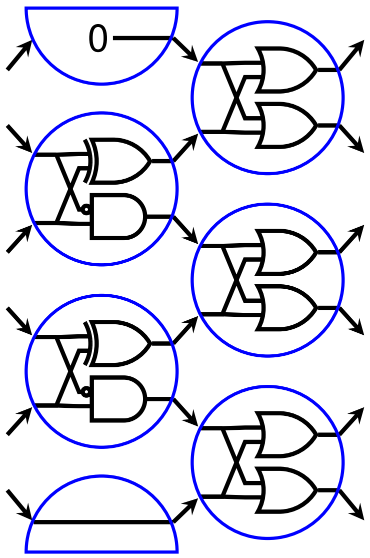
0 0 0 1 1 1



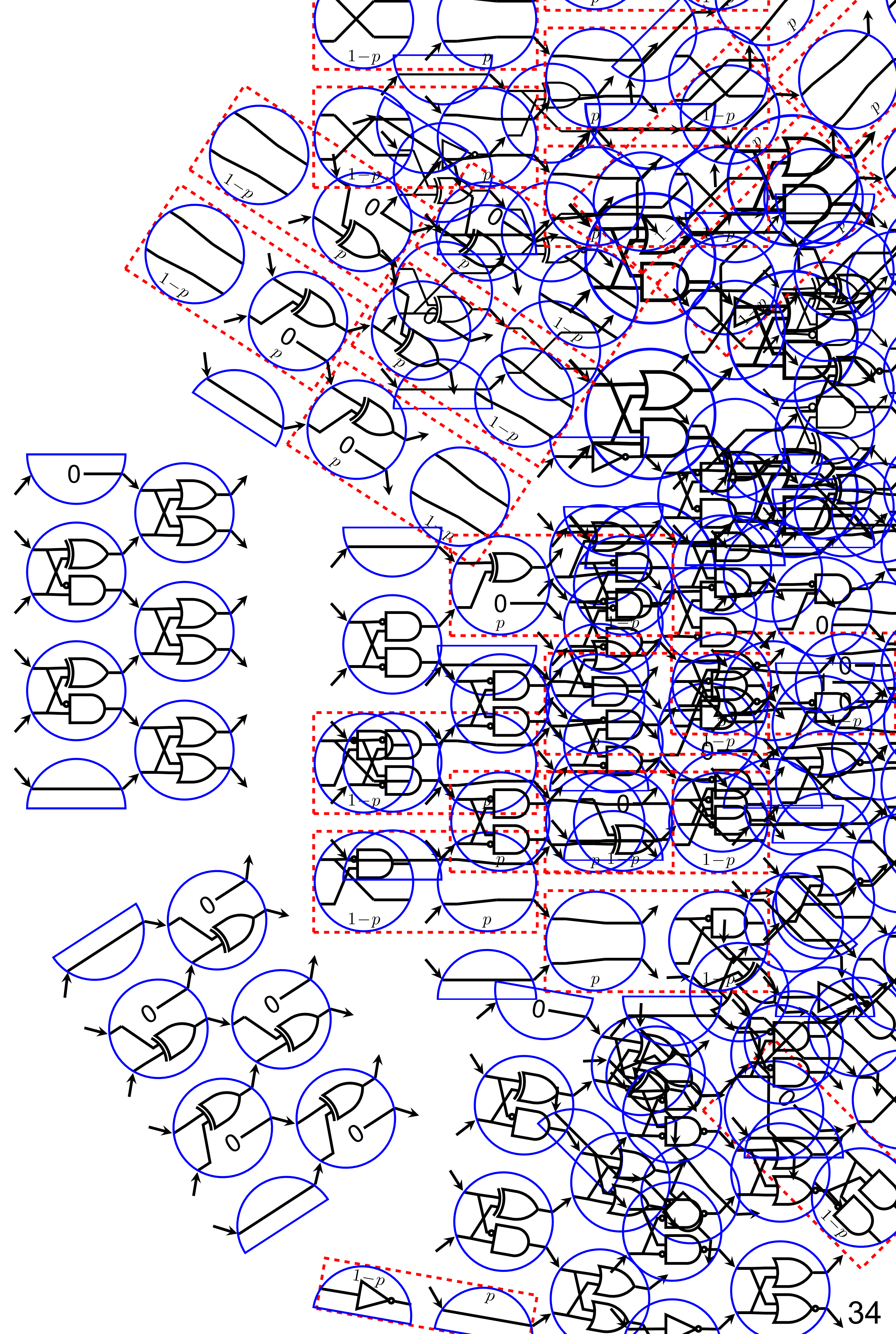
1 1 1 0 0 0

wall credit: Erik Klavins

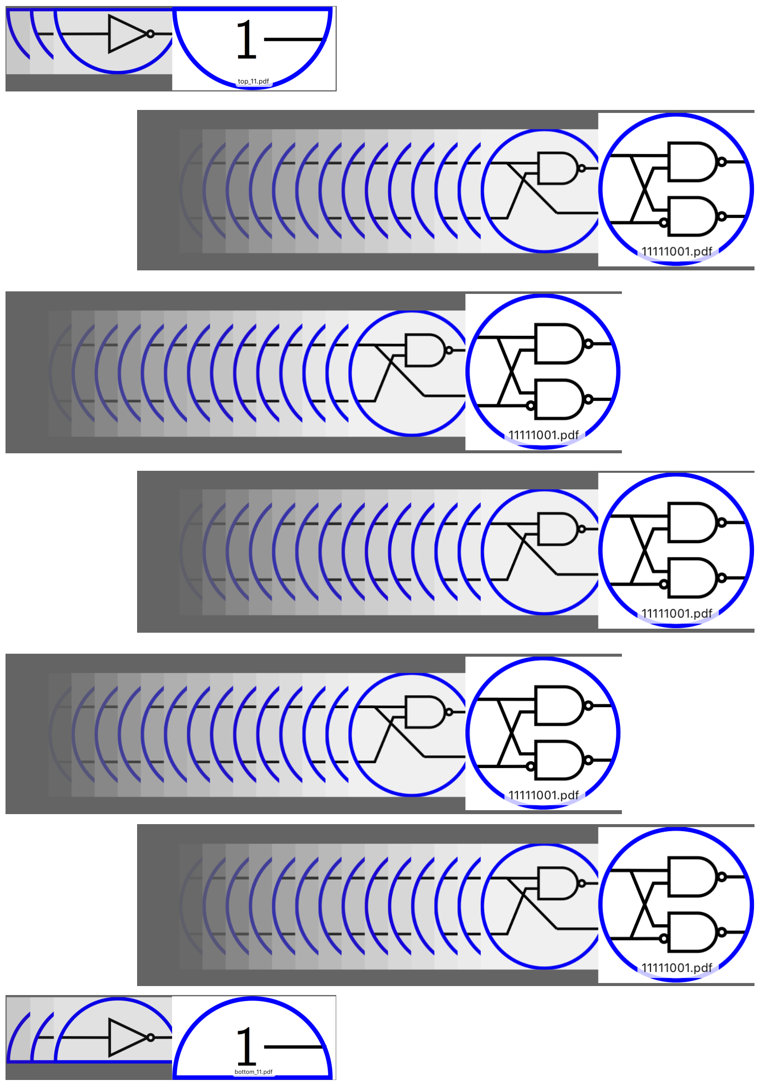
Which circuits to build?



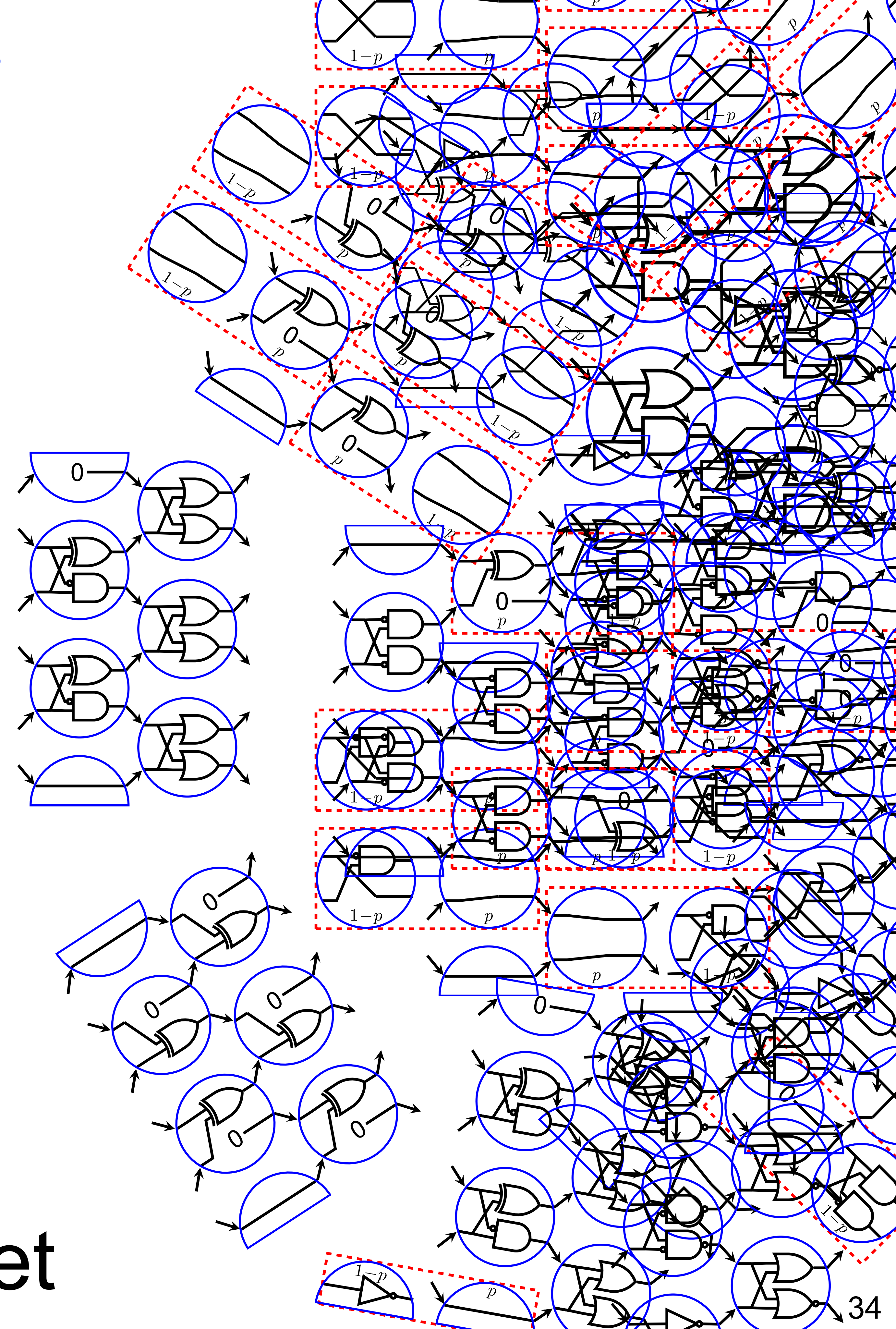
Which circuits to build?



Which circuits to build?



1,288 gates that implement **any** 6-bit circuit



“Complete” 6-bit gate set

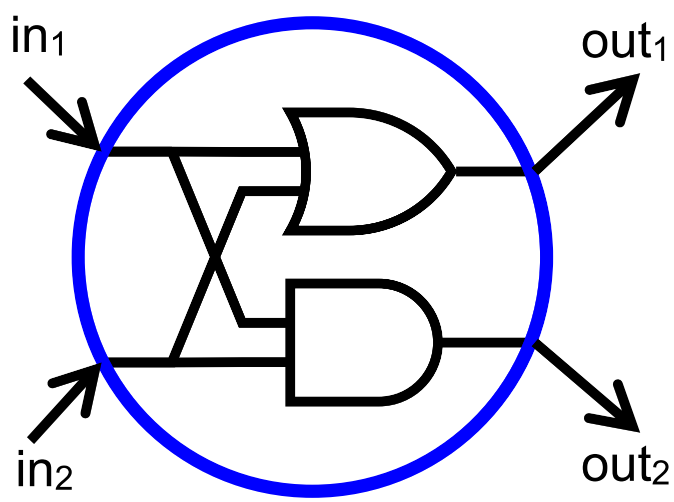
Structure

Theoretical circuit model

How it works: design and implementation

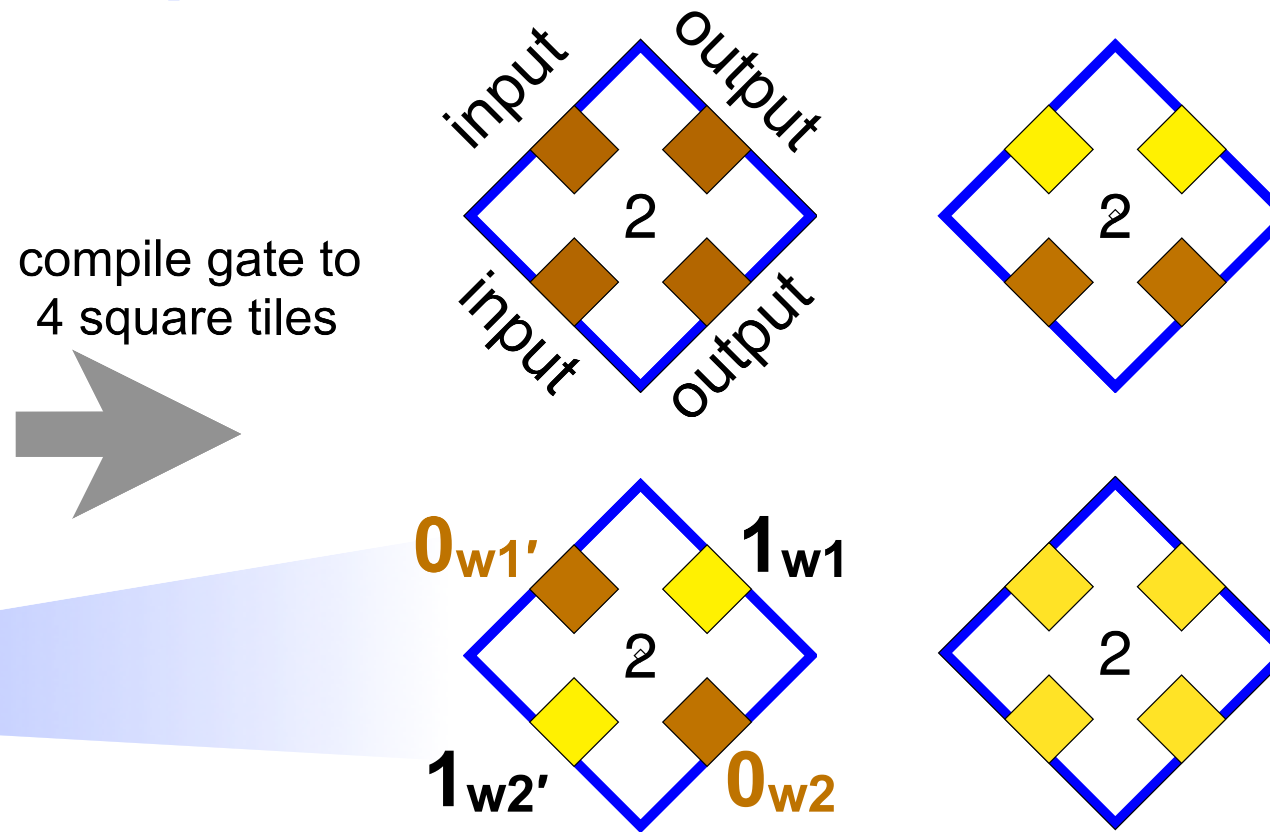
Experimental results

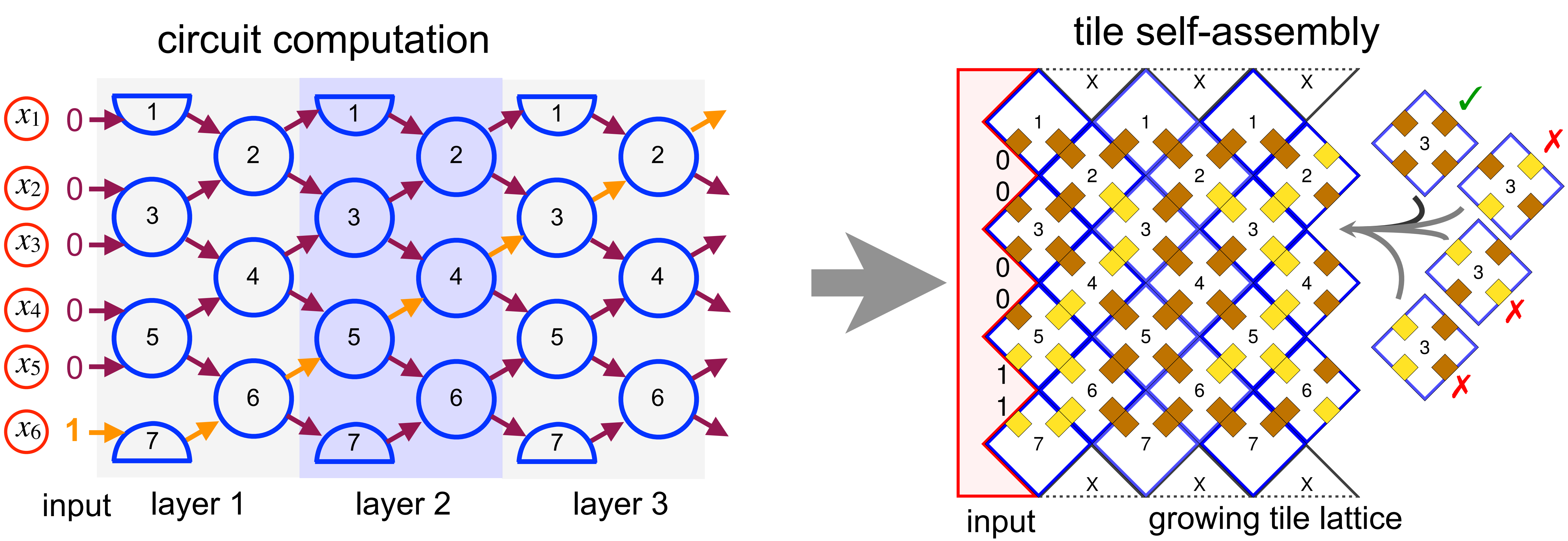
From circuits to square tiles



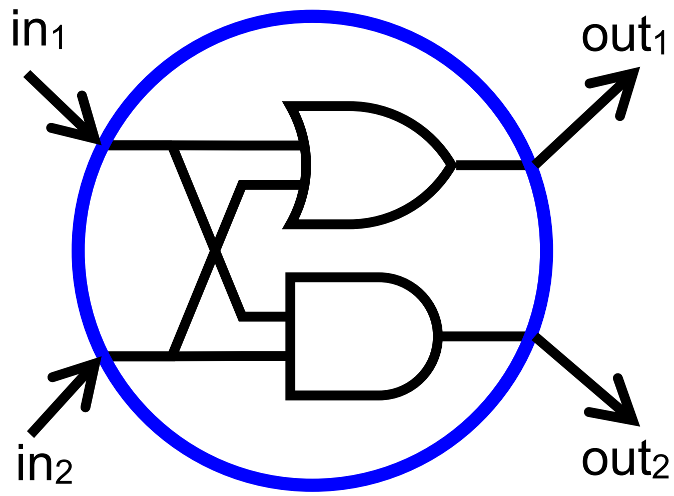
in1	in2	out1	out2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

gate truth table



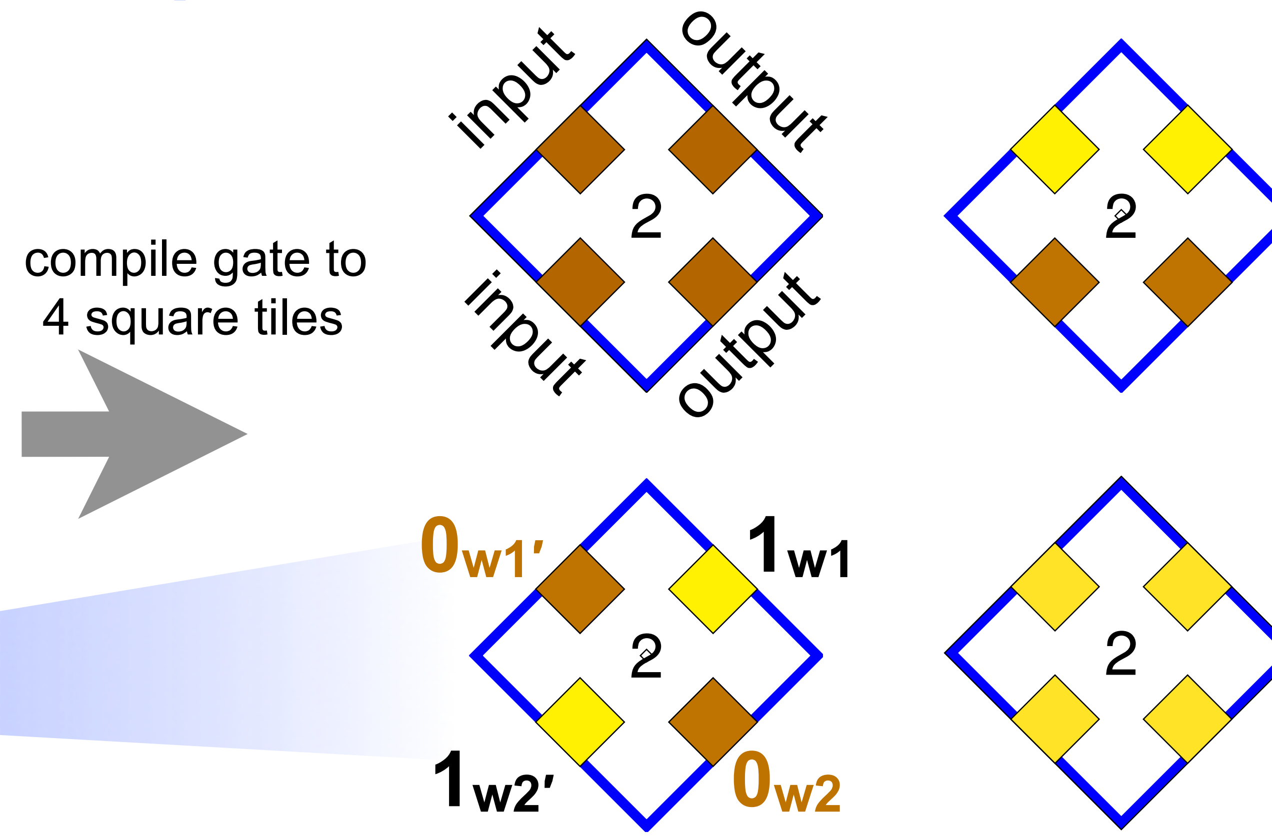


From circuits to square tiles

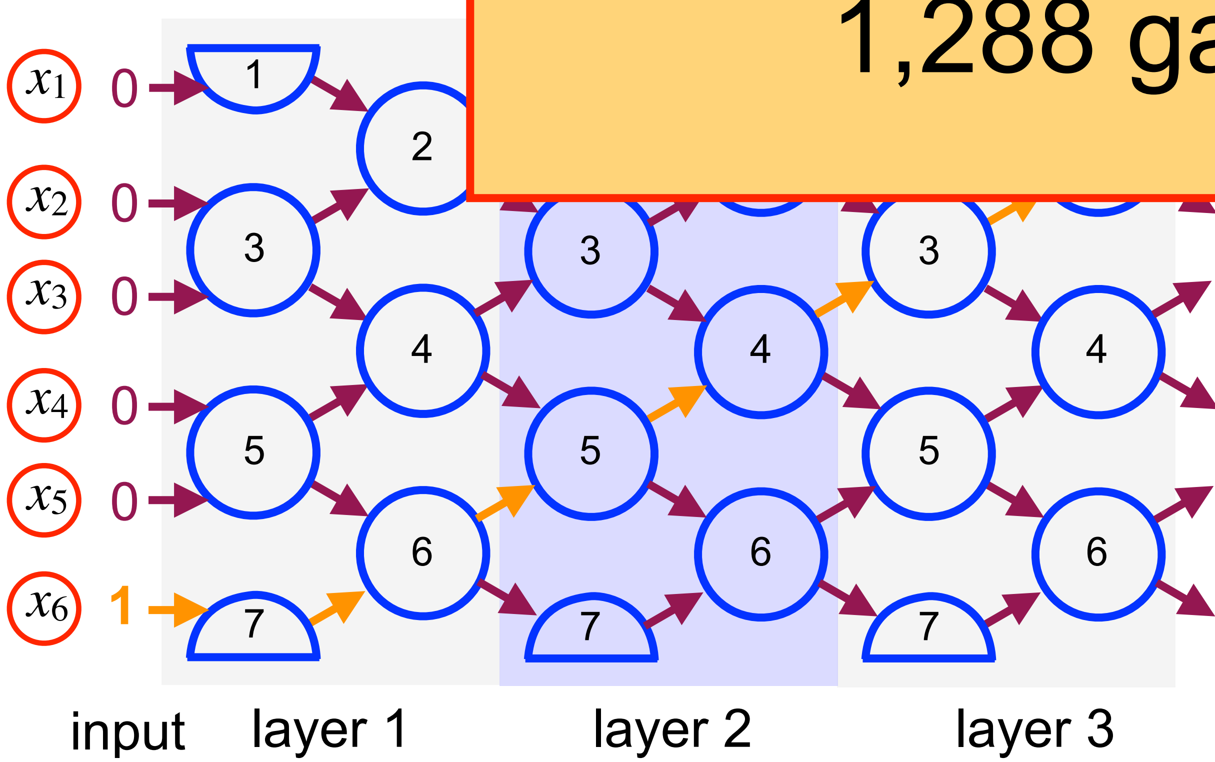


in_1	in_2	out_1	out_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

gate truth table

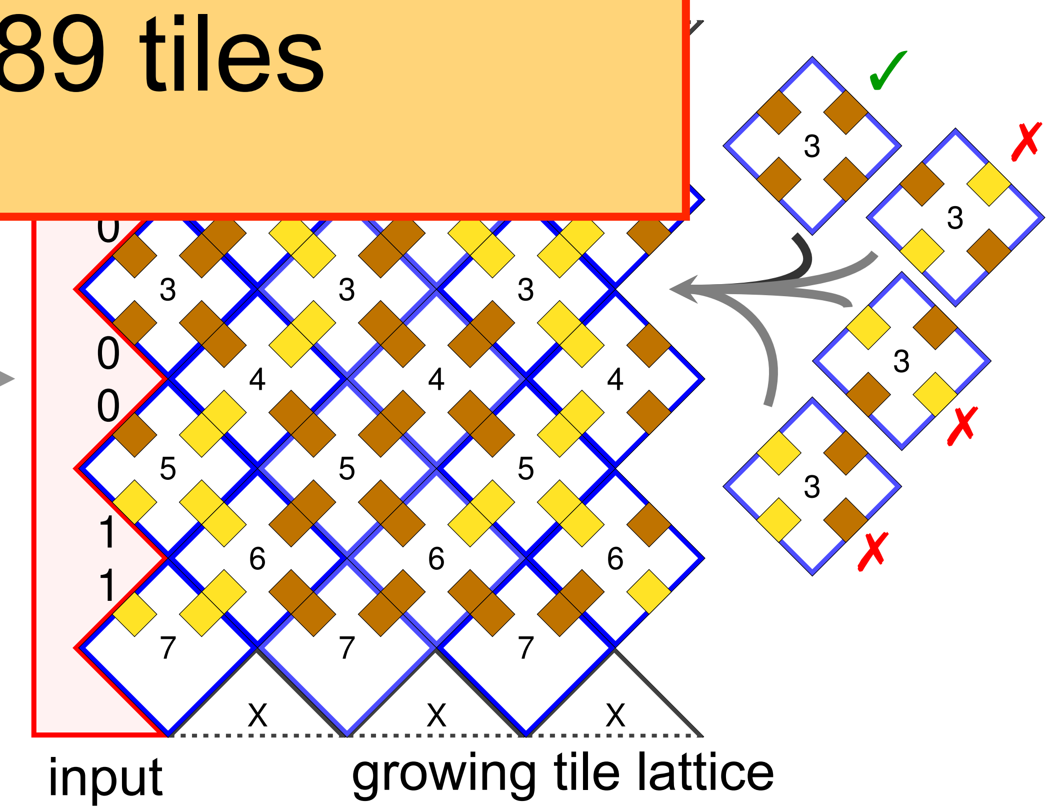


circuit c

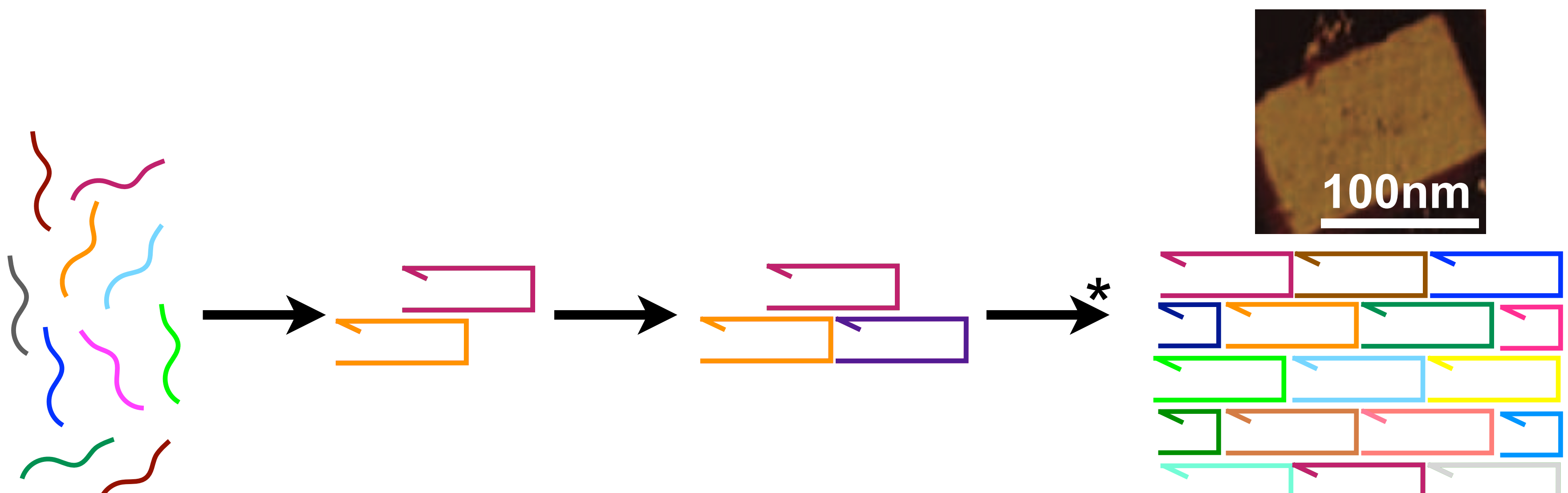


1,288 gates → 89 tiles

ly

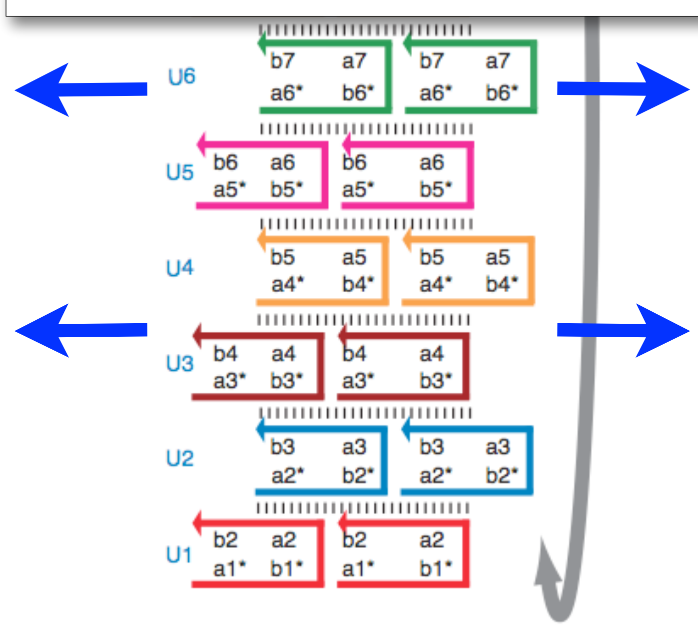


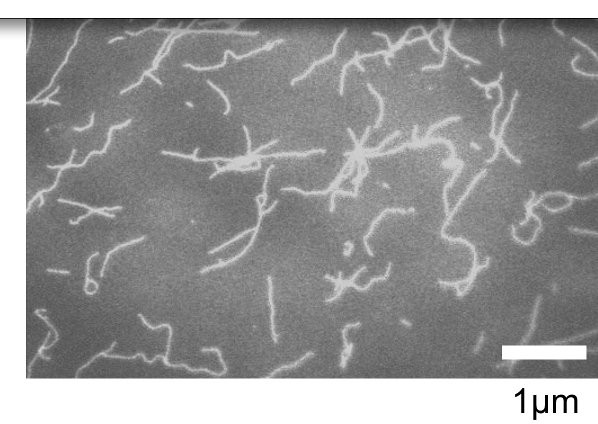
Single-stranded tile motif



DNA single-stranded tiles

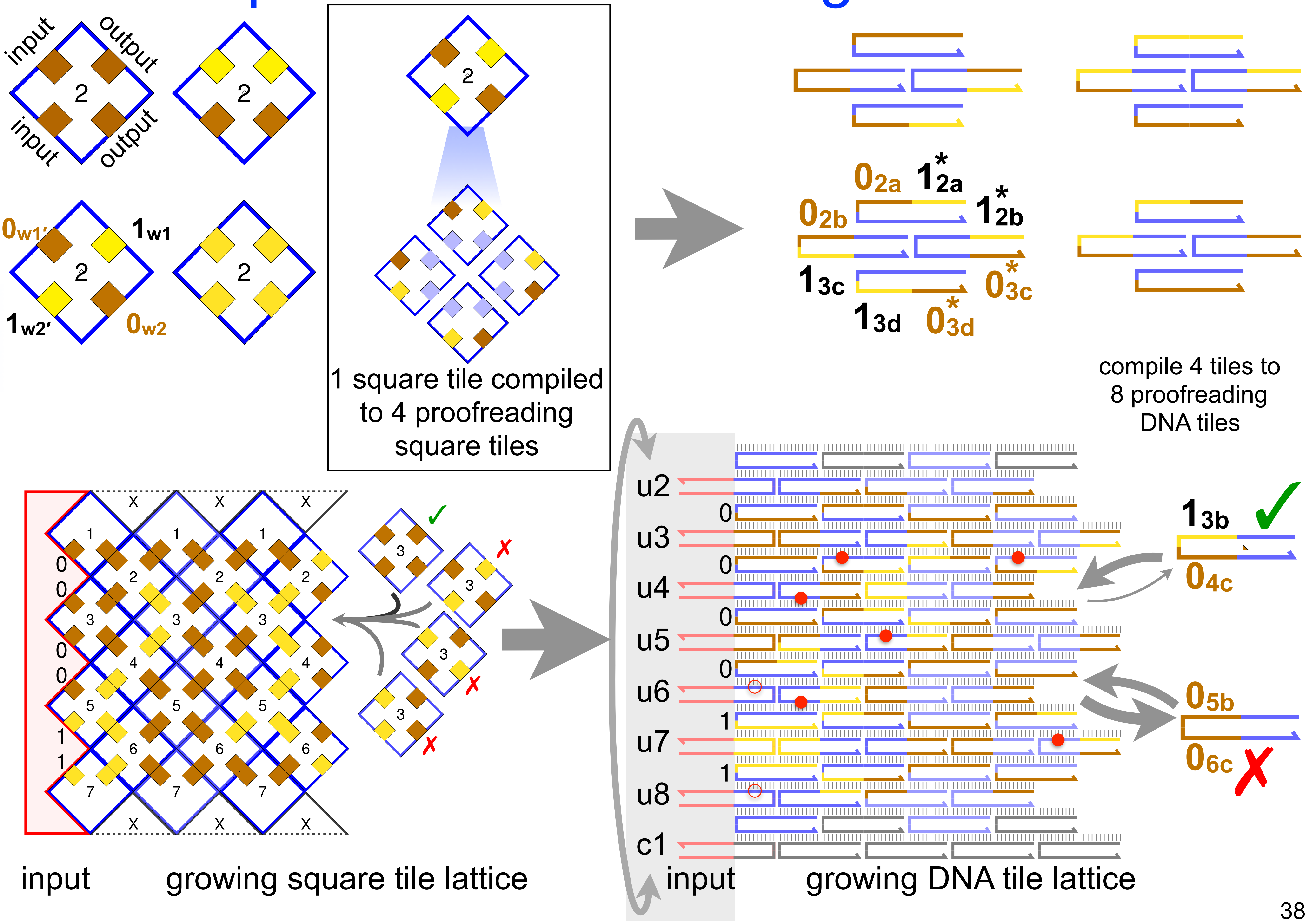
Wei, Dai, Yin. 2013 Nature



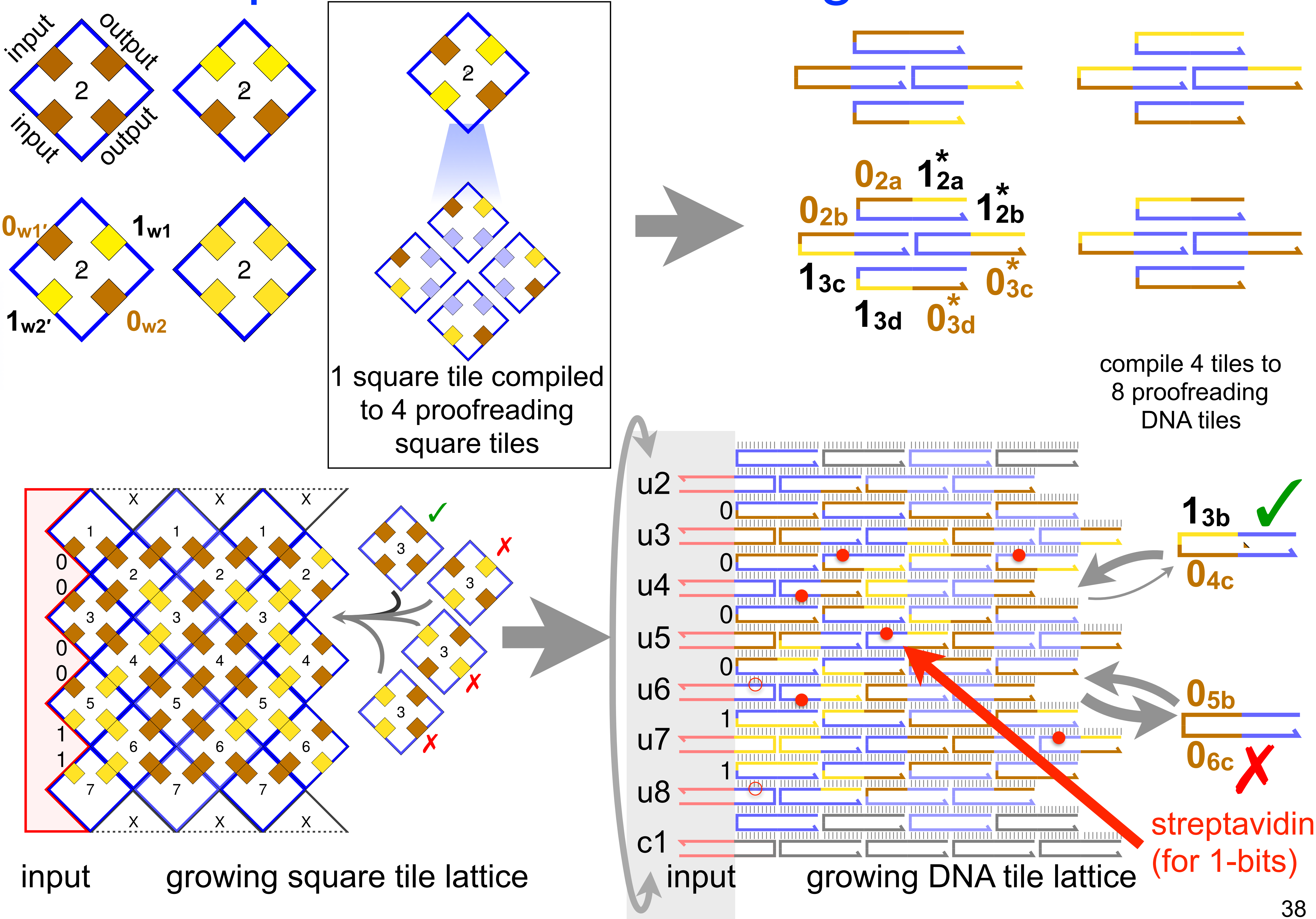


Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif. Science. 2008

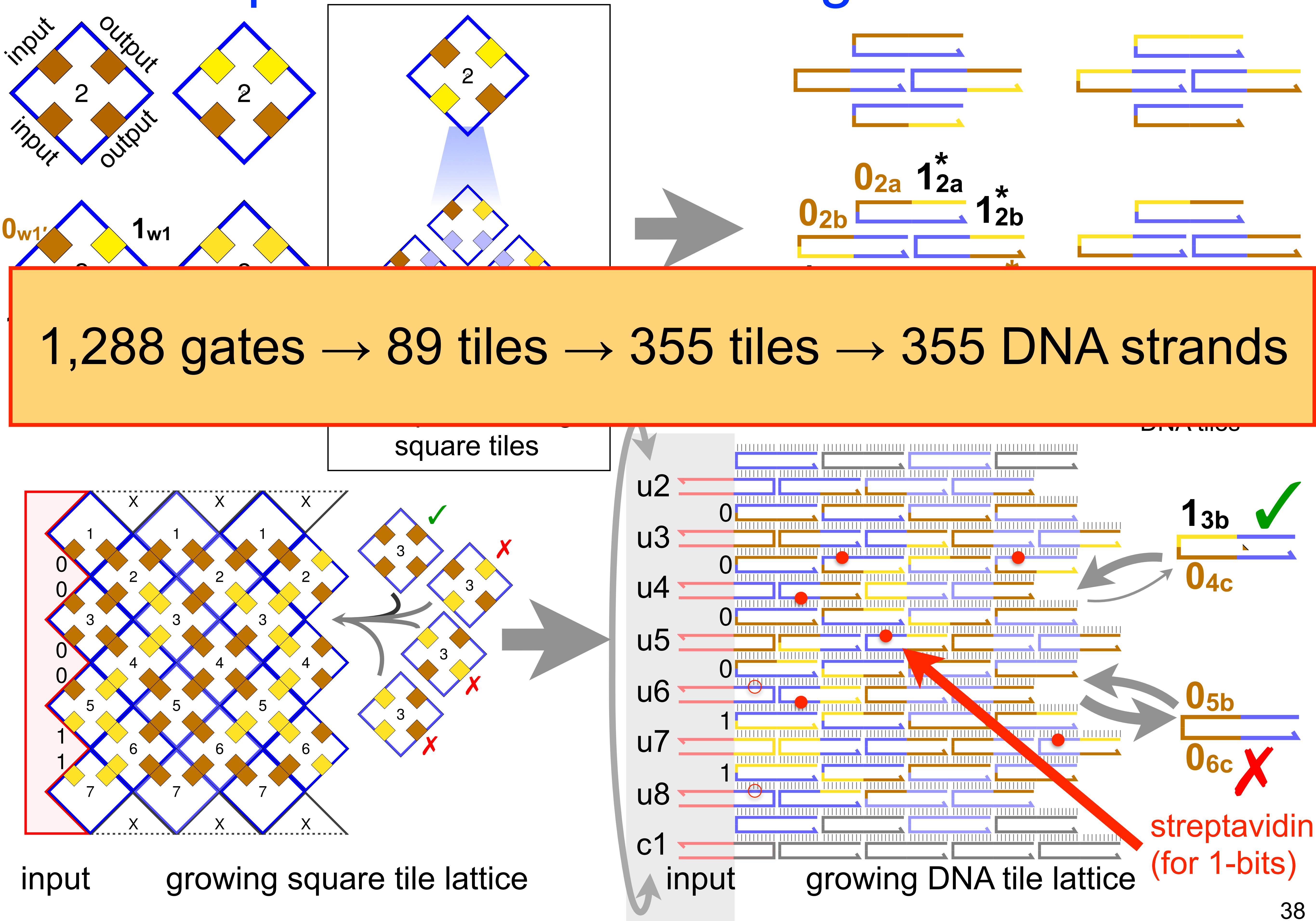
From square tiles to DNA single-stranded tiles



From square tiles to DNA single-stranded tiles

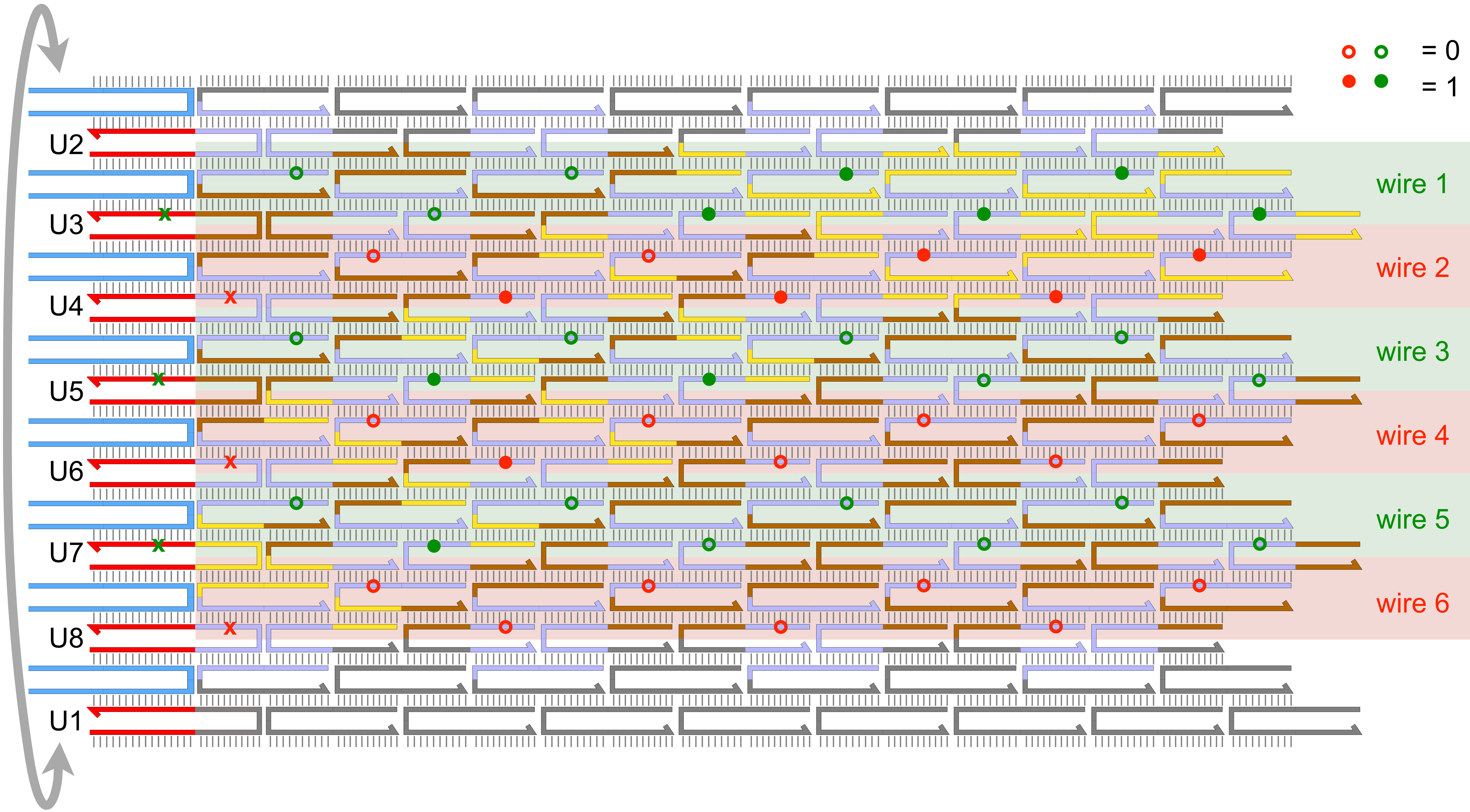


From square tiles to DNA single-stranded tiles



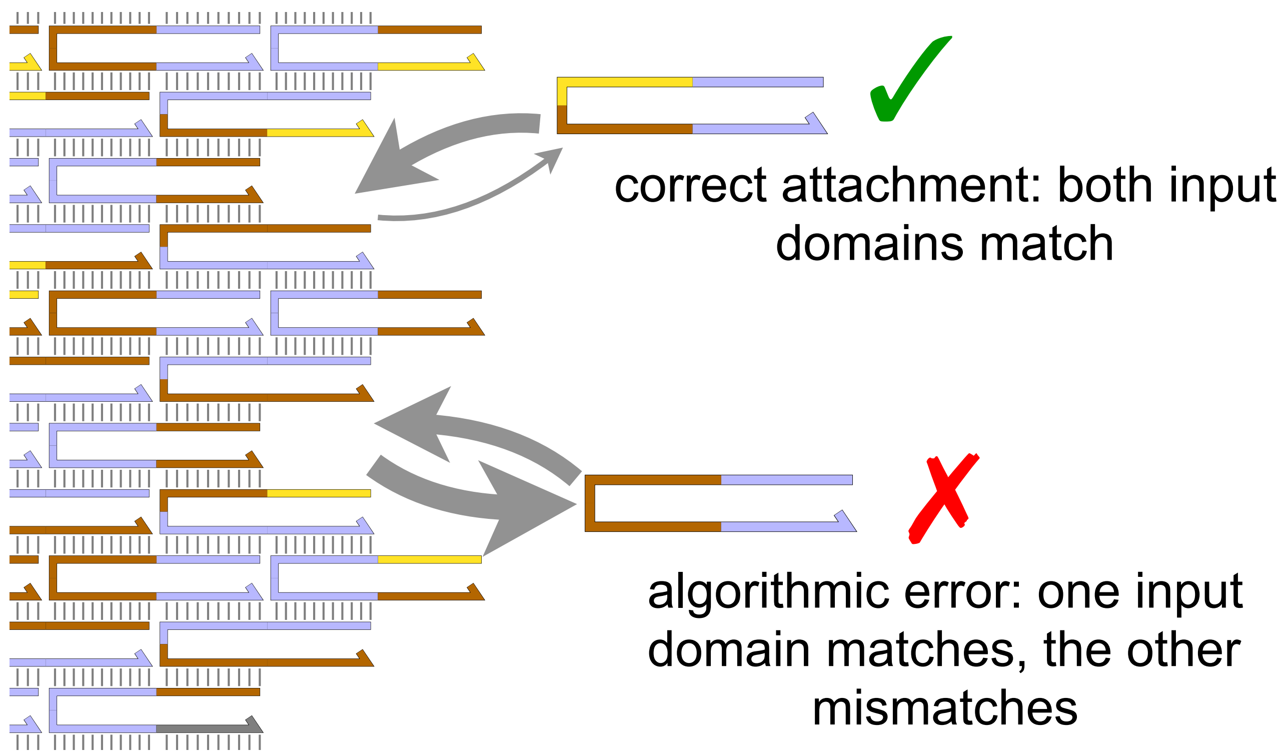
1,288 gates → 89 tiles → 355 tiles → 355 DNA strands

How the DNA tile lattice should look



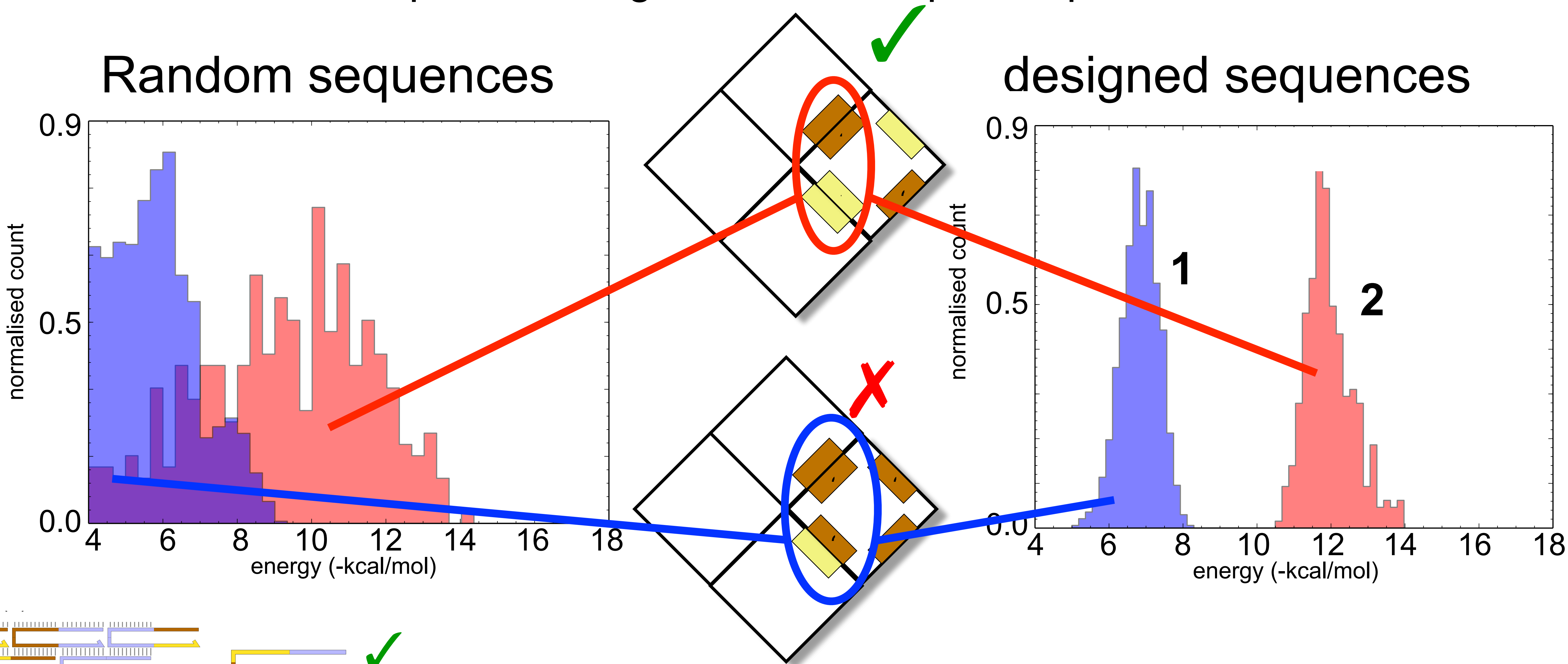
DNA sequence design

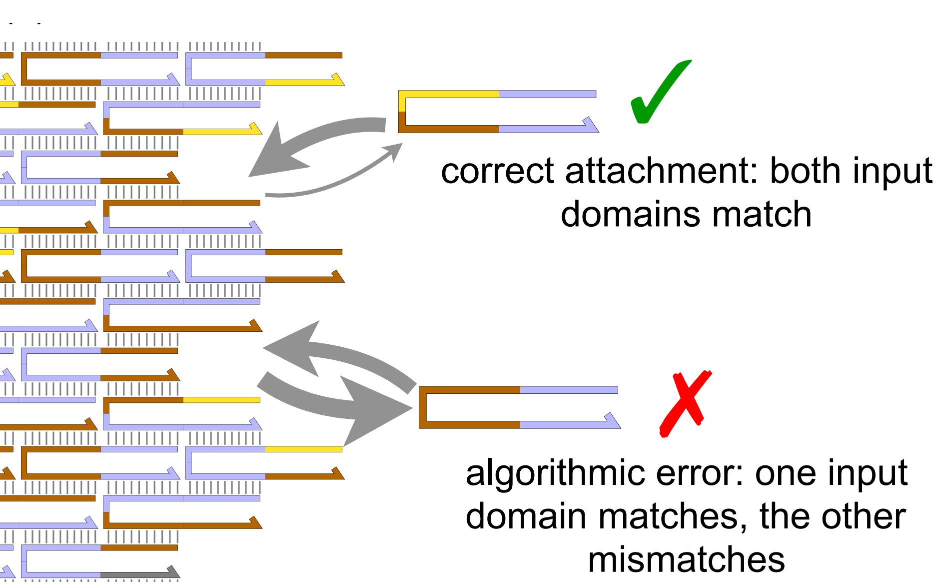
- Major challenge: We need to design DNA strands that bind when they should, and to not bind when the shouldn’t
- The next slide summarises some of the issues

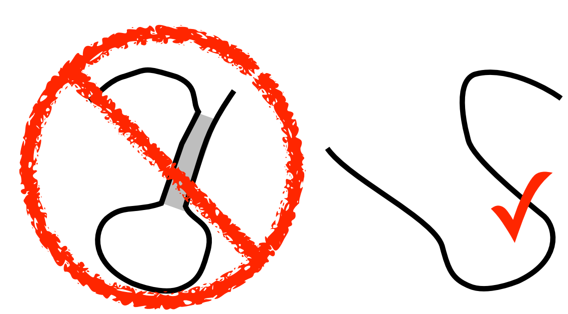


DNA sequence design

- Major challenge: We need to design DNA strands that bind when they should, and to not bind when they shouldn't
- Custom DNA sequence designer; built on top of Nupack & ViennaRNA

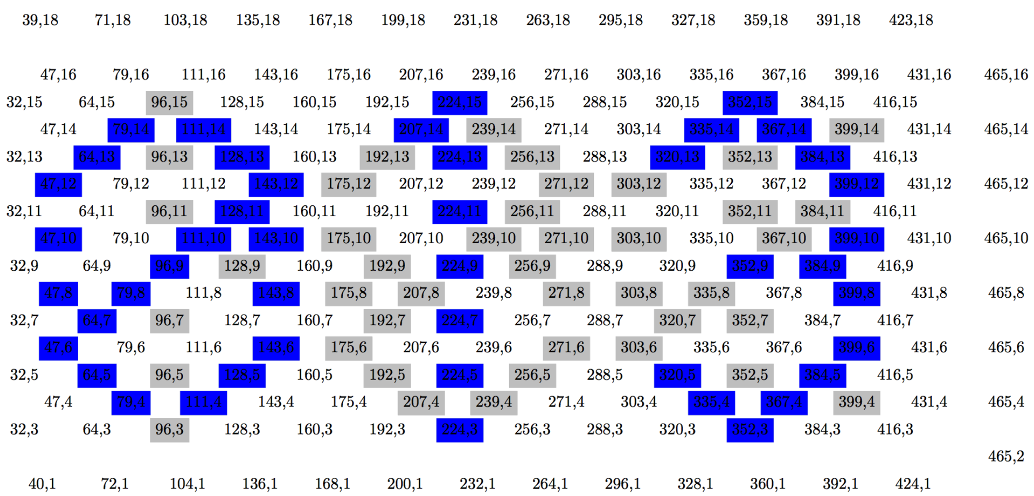






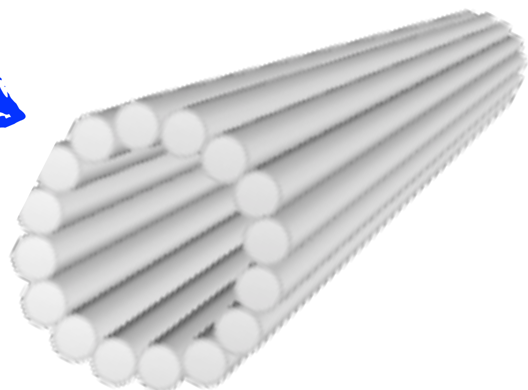
- **3. Strand ss**
- **4. Clean lattice**
- **5. Strand pairs**

Barcoded DNA origami seed

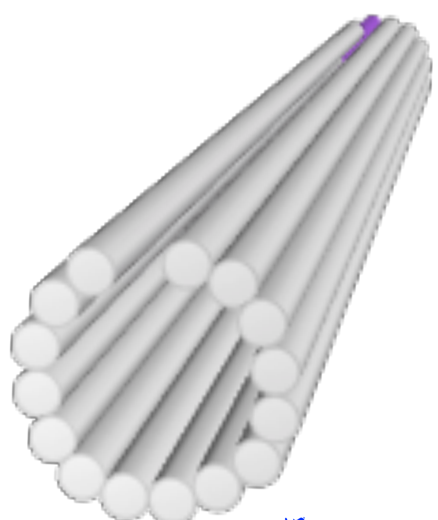


Choose which staples have biotin modifications

Form 16-helix tube

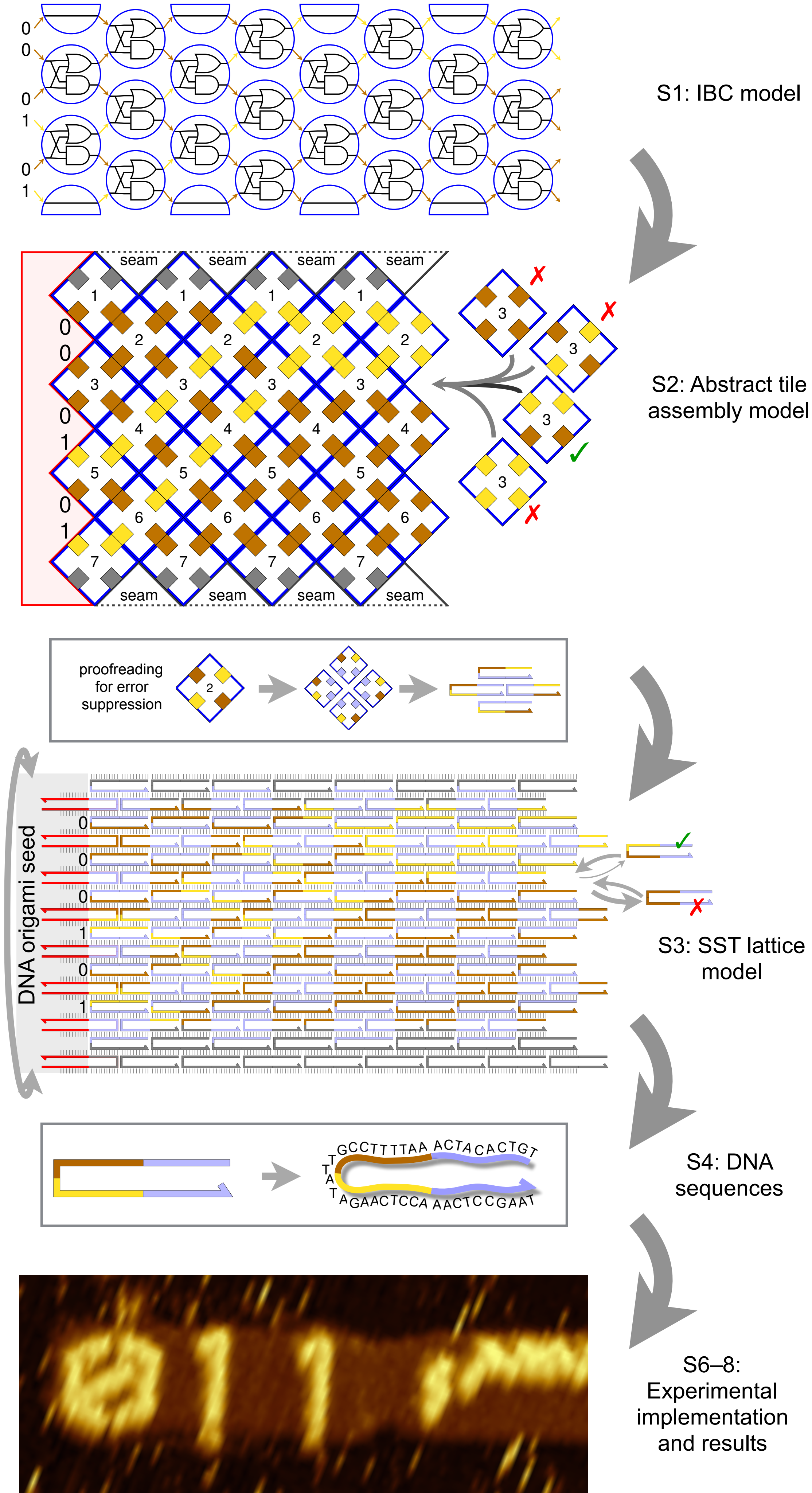


Unzip



add streptadividin & image on mica

Abstraction hierarchy



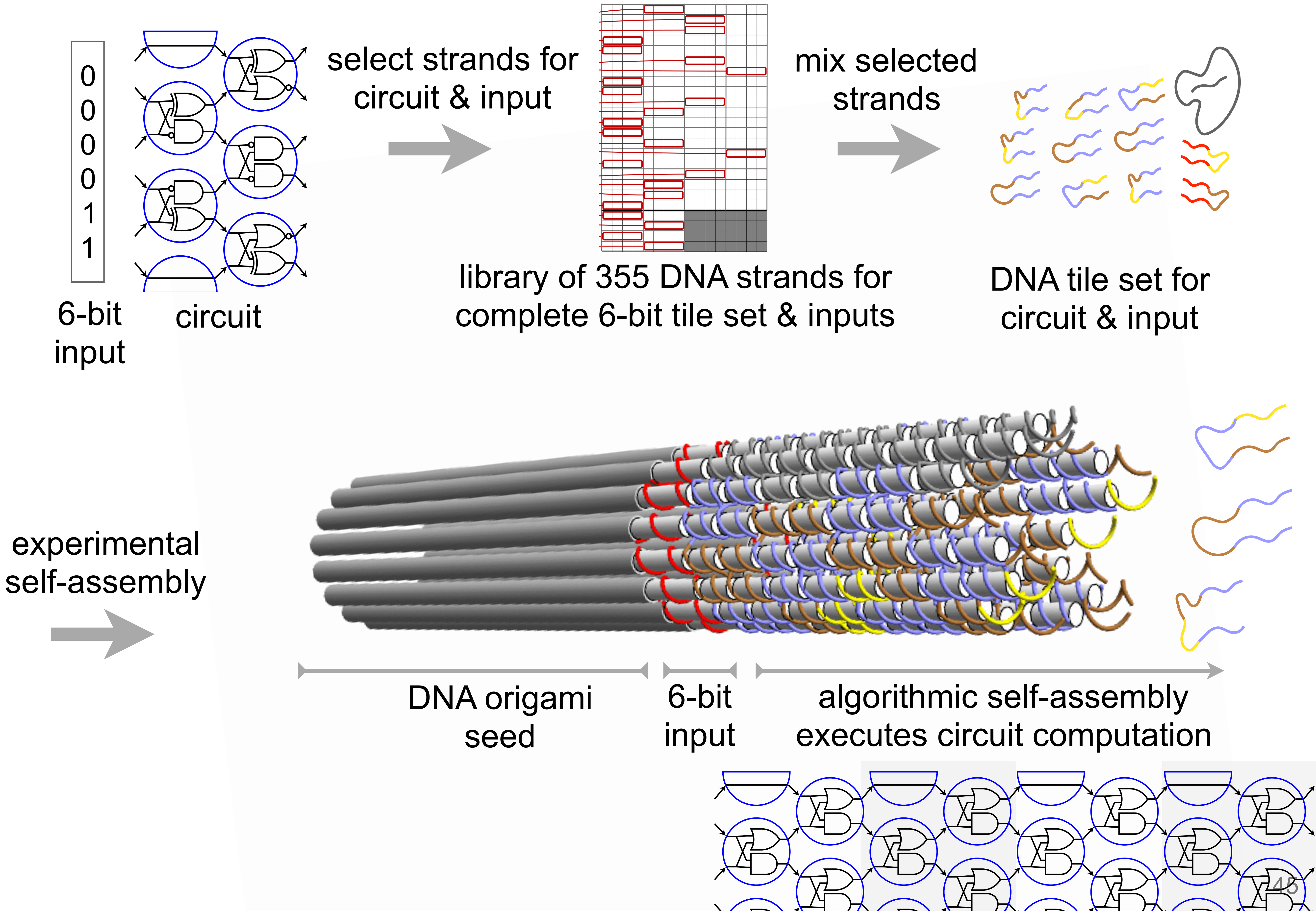
Structure

Theoretical circuit model

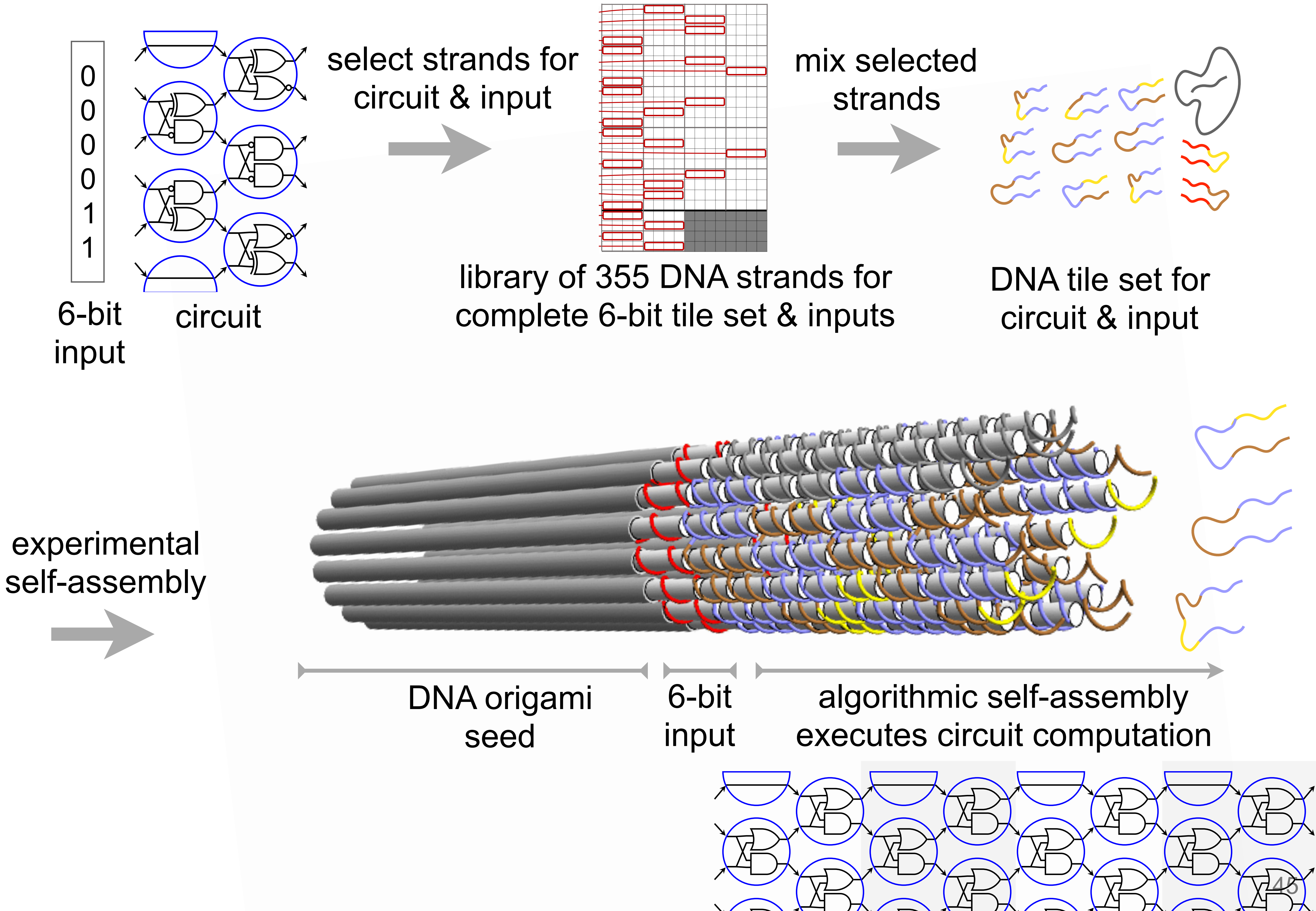
How it works: design and implementation

Experimental results

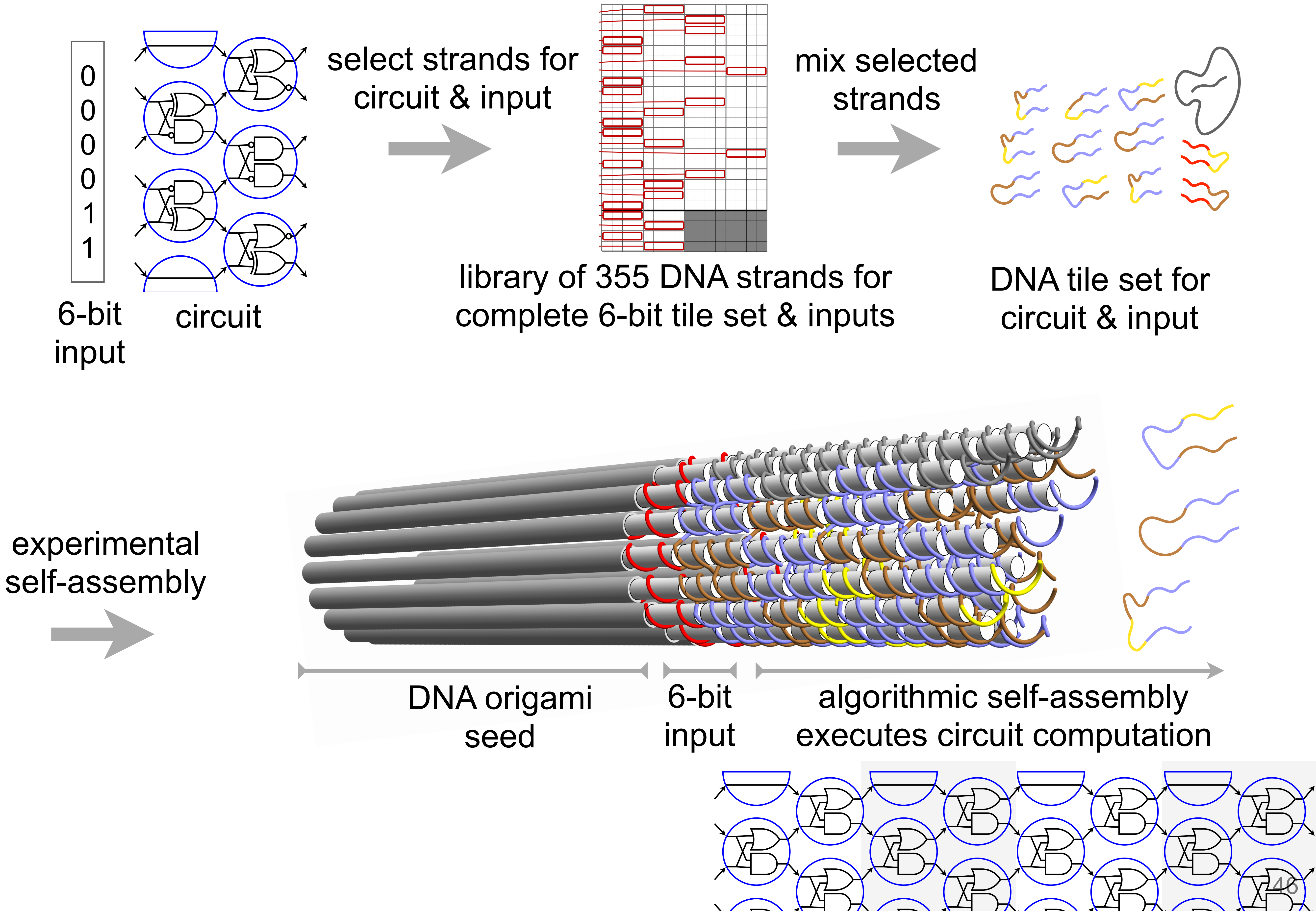
Schematic


0
0
0
0
1
1
6-bit input
circuit
select strands for circuit & input
library of 355 DNA strands for complete 6-bit tile set & inputs
mix selected strands
DNA tile set for circuit & input
experimental self-assembly
DNA origami seed
6-bit input
algorithmic self-assembly executes circuit computation

Schematic

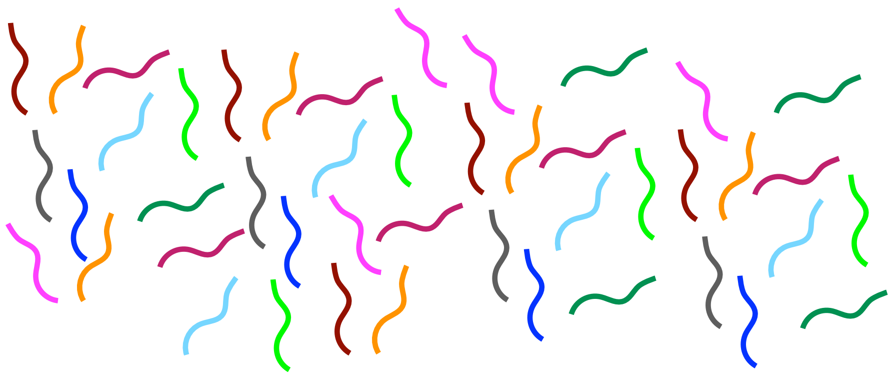

0
0
0
0
1
1
6-bit input
circuit
select strands for circuit & input
library of 355 DNA strands for complete 6-bit tile set & inputs
mix selected strands
DNA tile set for circuit & input
experimental self-assembly
DNA origami seed
6-bit input
algorithmic self-assembly executes circuit computation

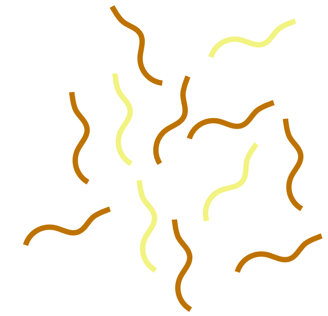
Schematic


0
0
0
0
1
1
6-bit input
circuit
select strands for circuit & input
library of 355 DNA strands for complete 6-bit tile set & inputs
mix selected strands
DNA tile set for circuit & input
experimental self-assembly
DNA origami seed
6-bit input
algorithmic self-assembly executes circuit computation

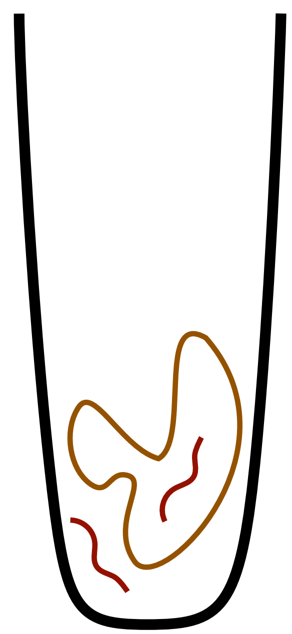
An example experiment: Sorting

355 tiles that implement **any 6-bit circuit**





6-bit **input** strands

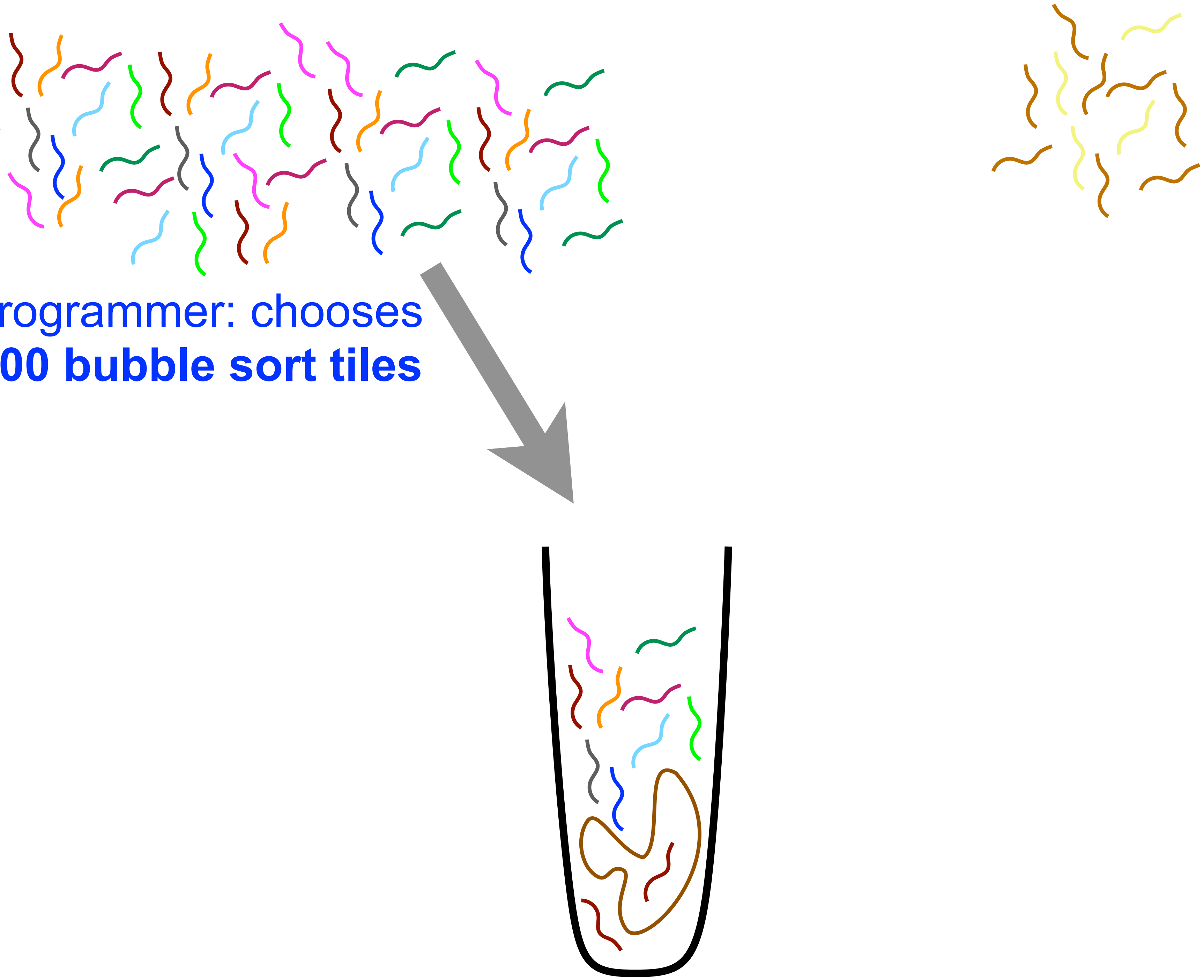


tiles & seed

An example experiment: SORTING

355 tiles that implement **any 6-bit circuit**

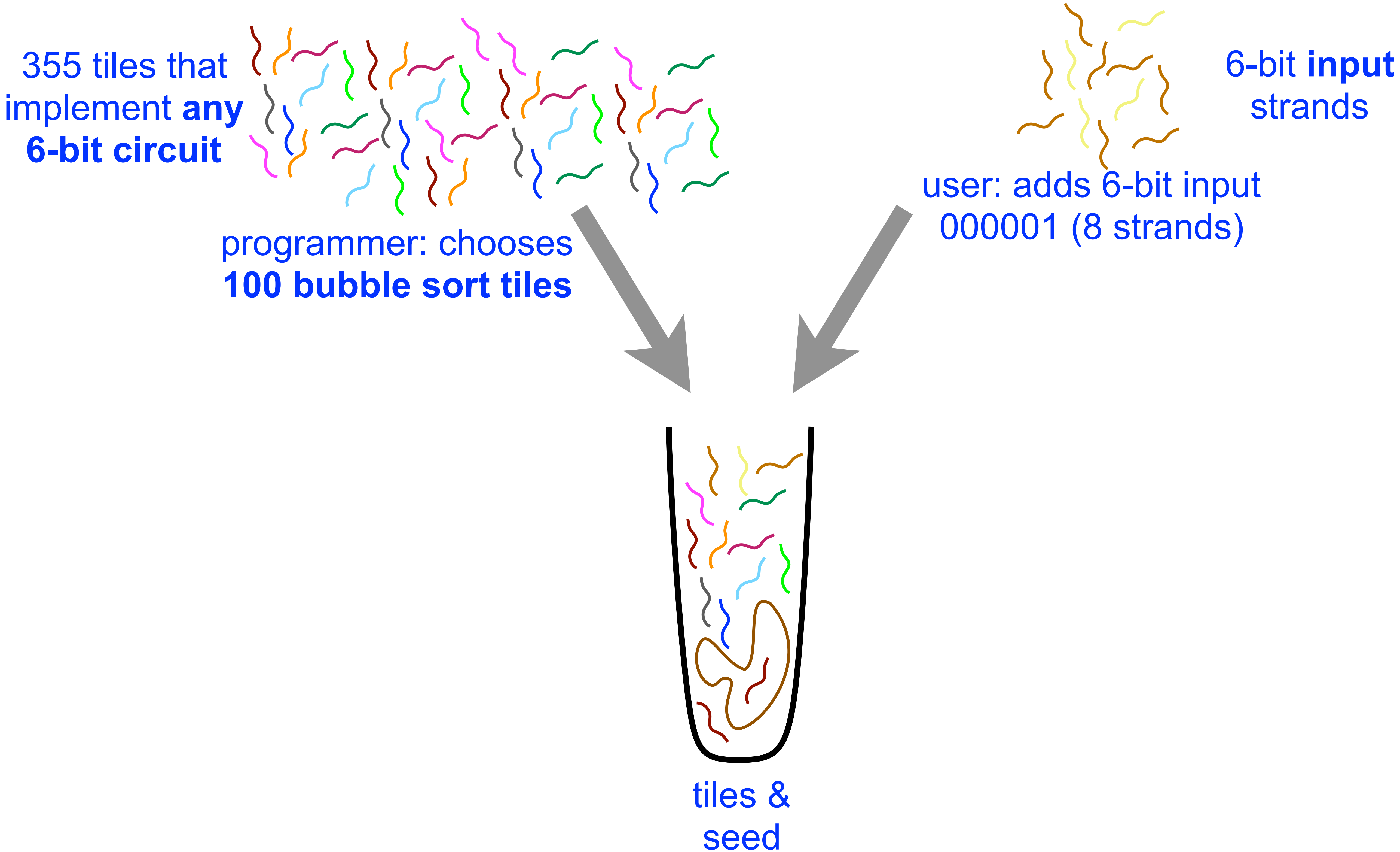
6-bit **input** strands



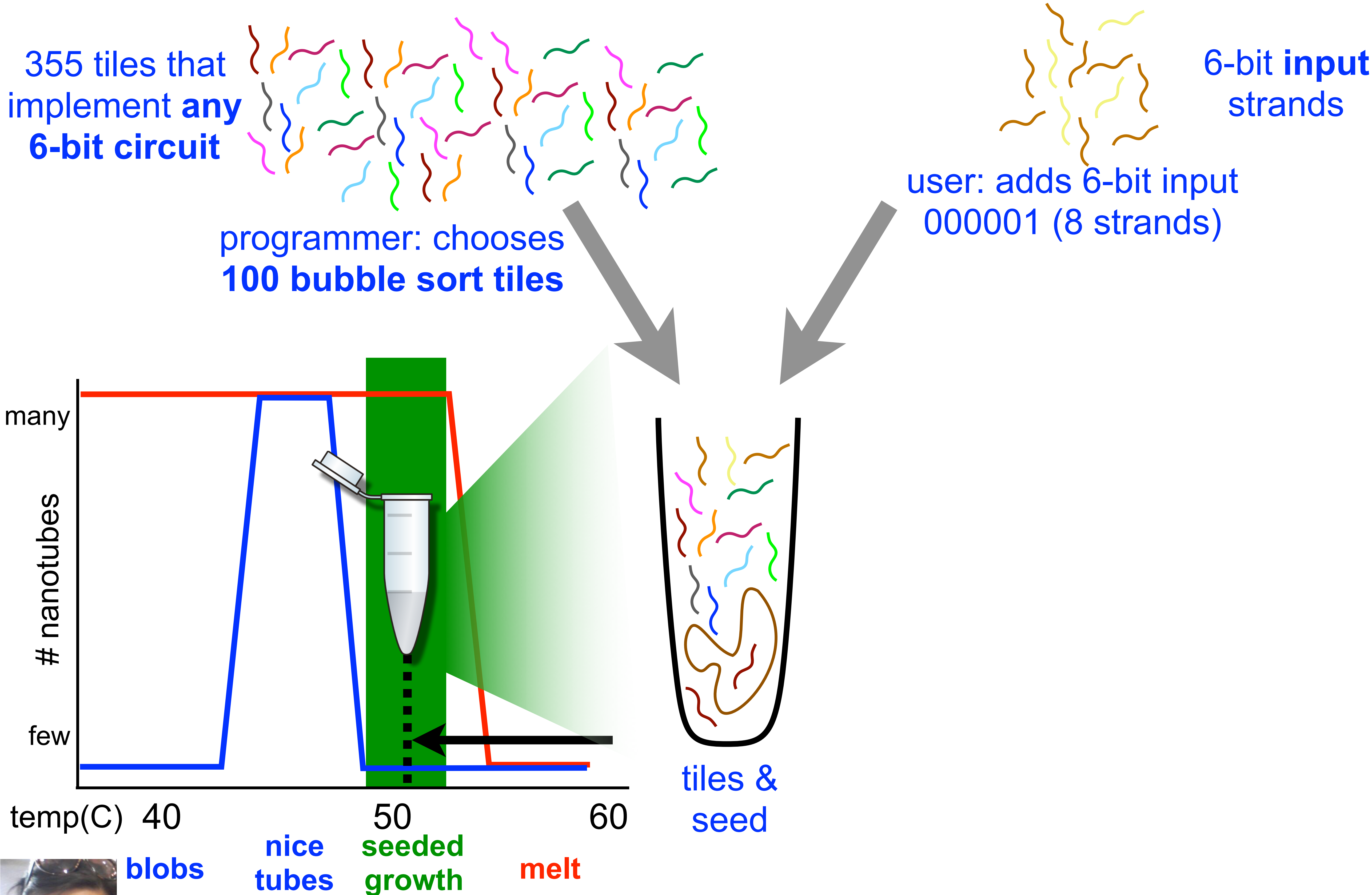
programmer: chooses **100 bubble sort tiles**

tiles & seed

An example experiment: SORTING

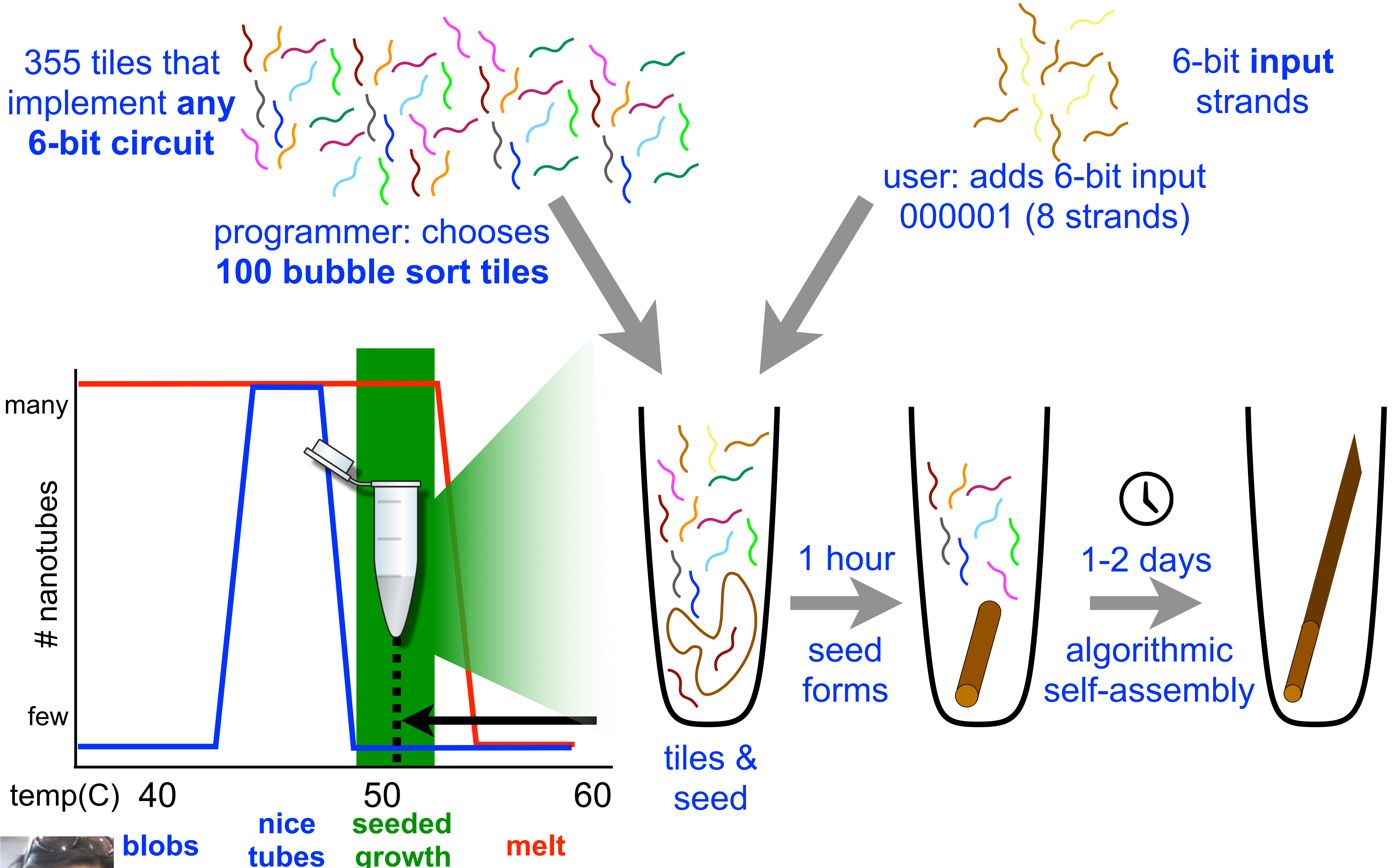


An example experiment: SORTING



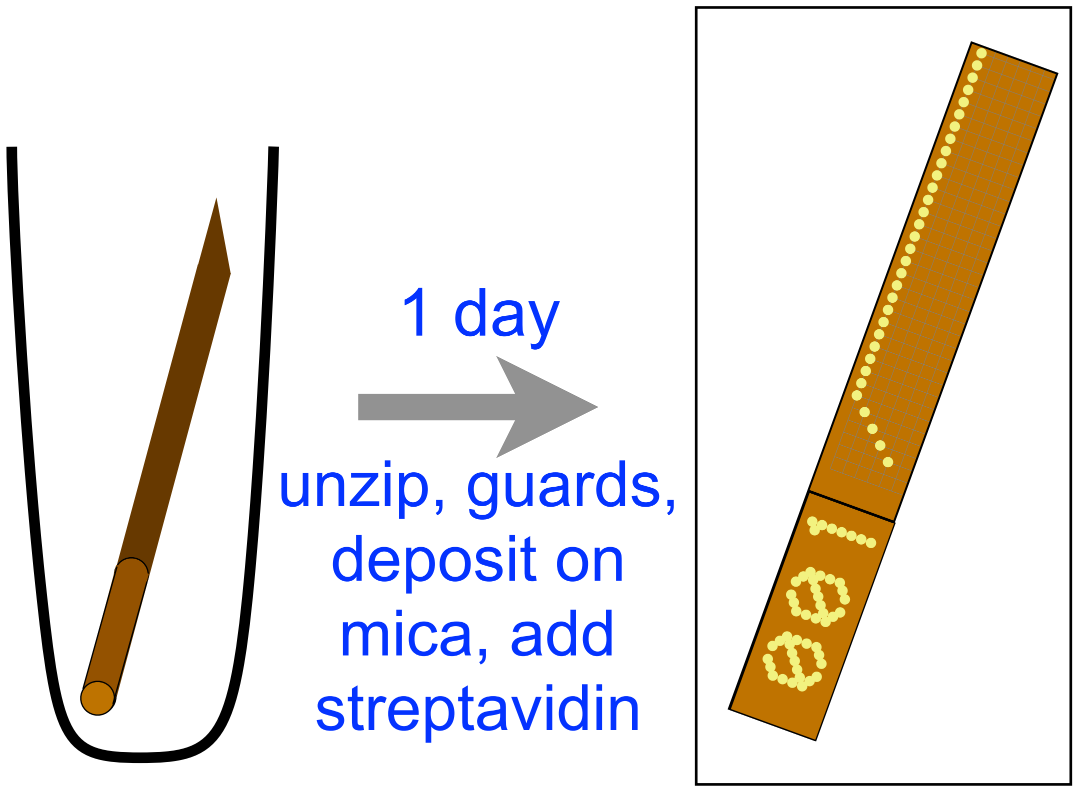
Joy Hui

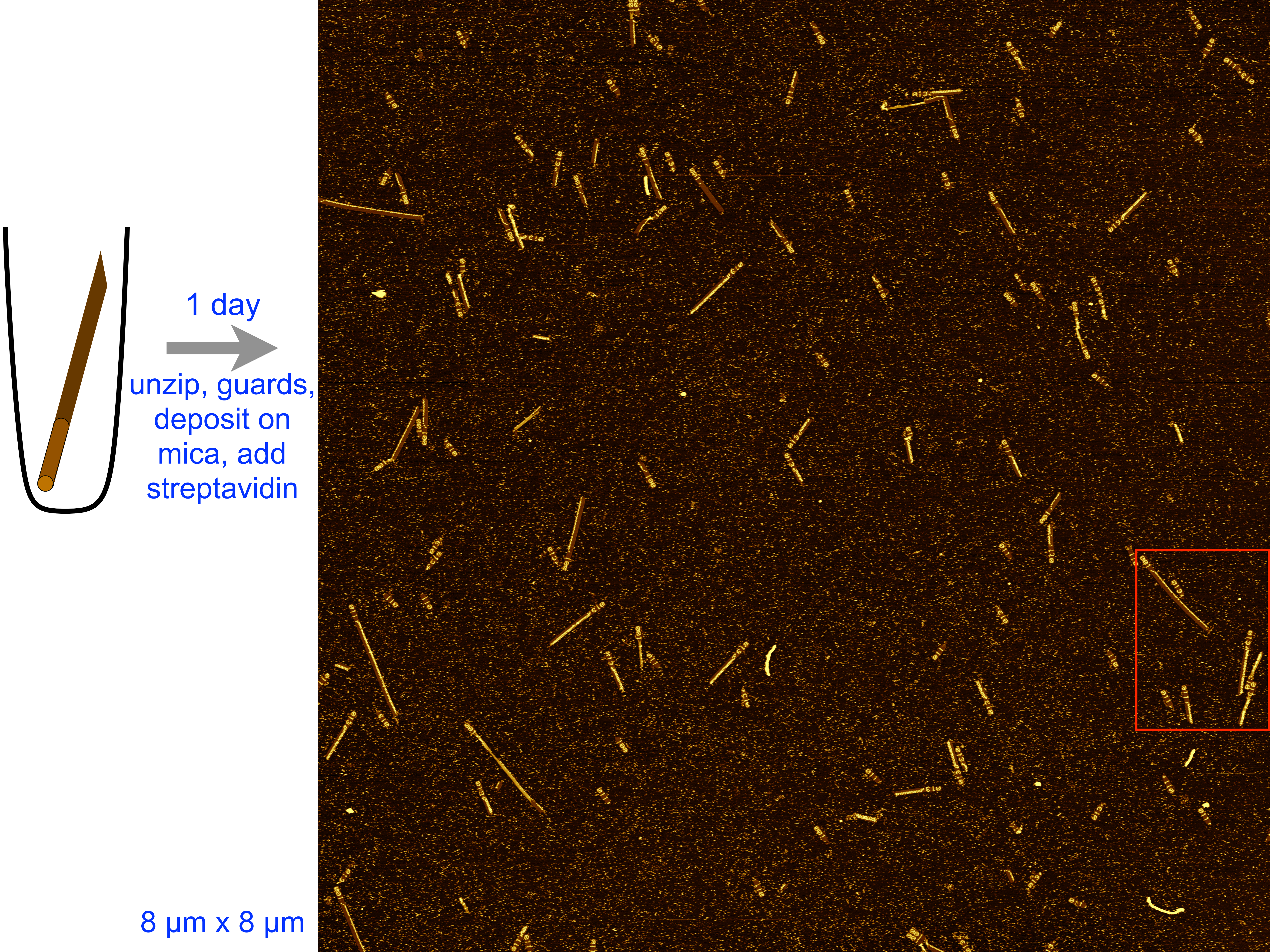
An example experiment: SORTING



Joy Hui

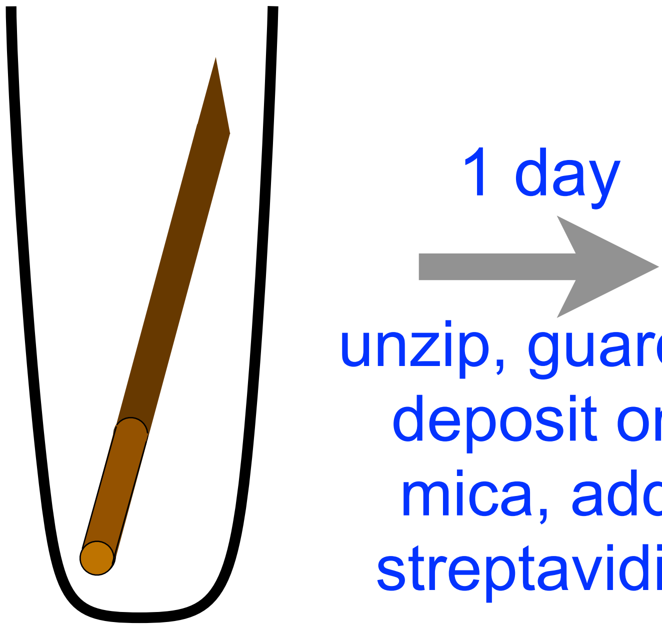
An example experiment: SORTING





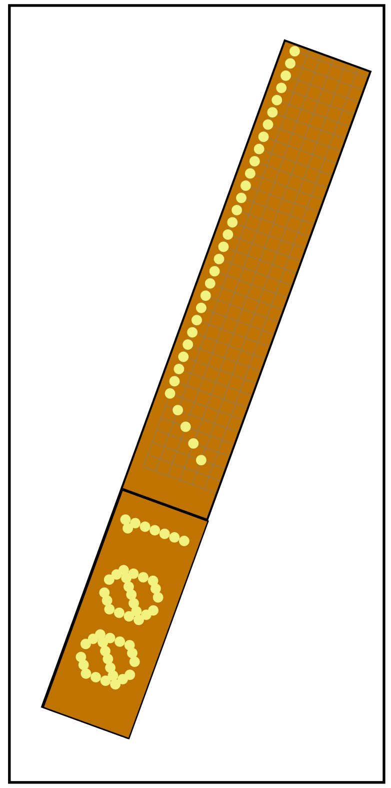
1 day
unzip, guards, deposit on mica, add streptavidin
8 µm x 8 µm

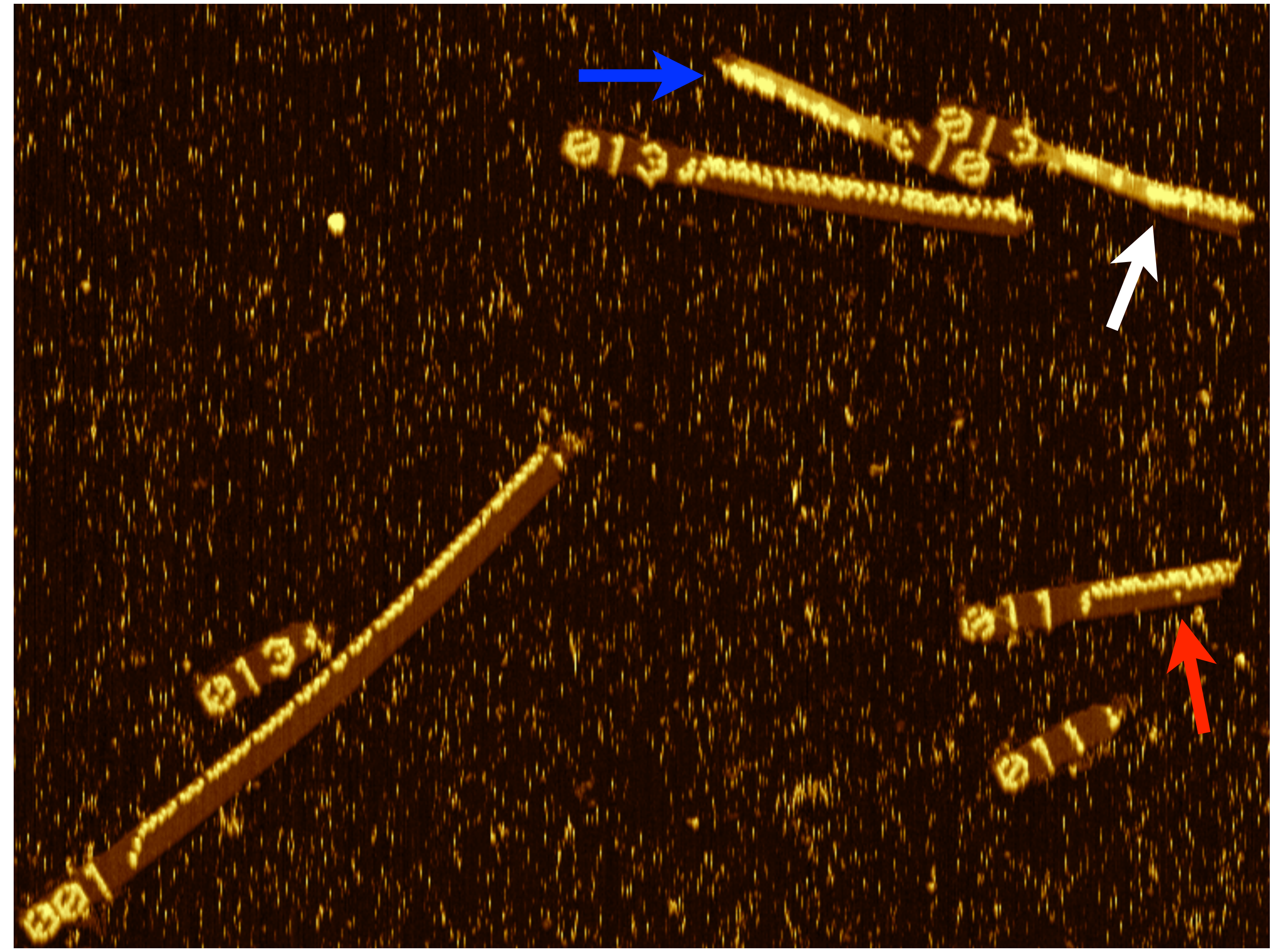
An example experiment: Sorting



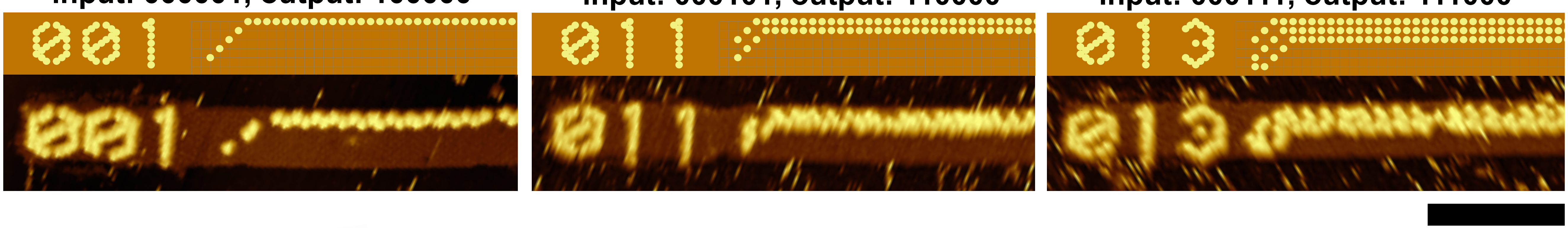
1 day

unzip, guards, deposit on mica, add streptavidin

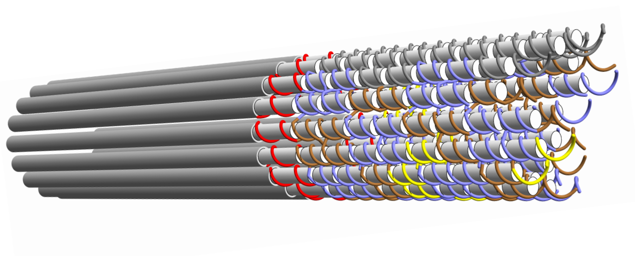




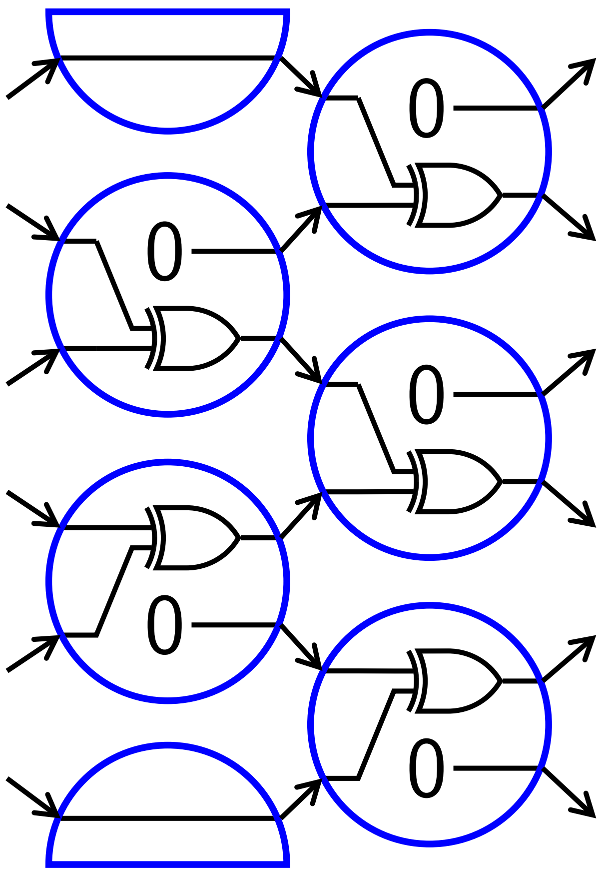
input: 000001, output: 100000 | input: 000101, output: 110000 | input: 000111, output: 111000

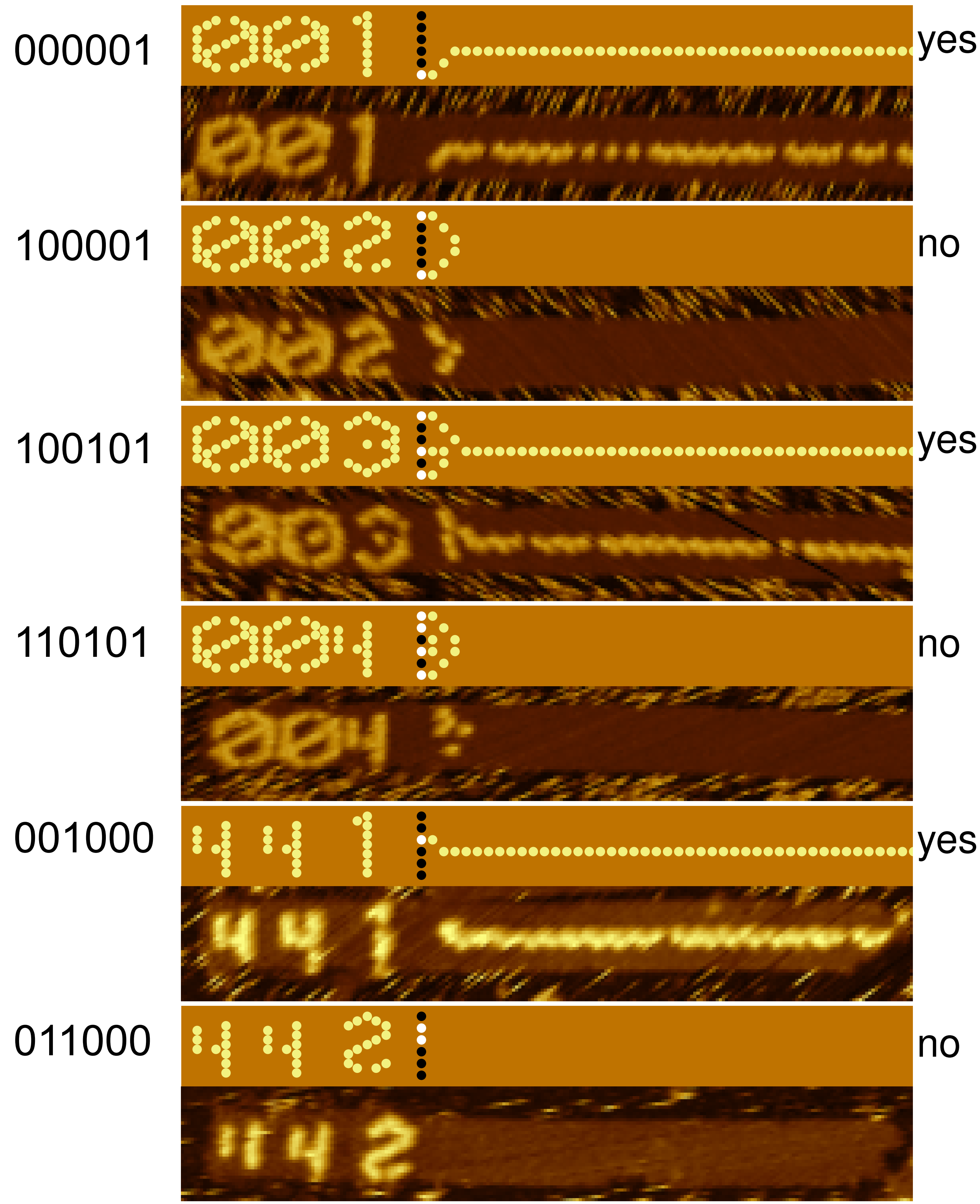


100nm



PARITY: is the number of 1s odd?

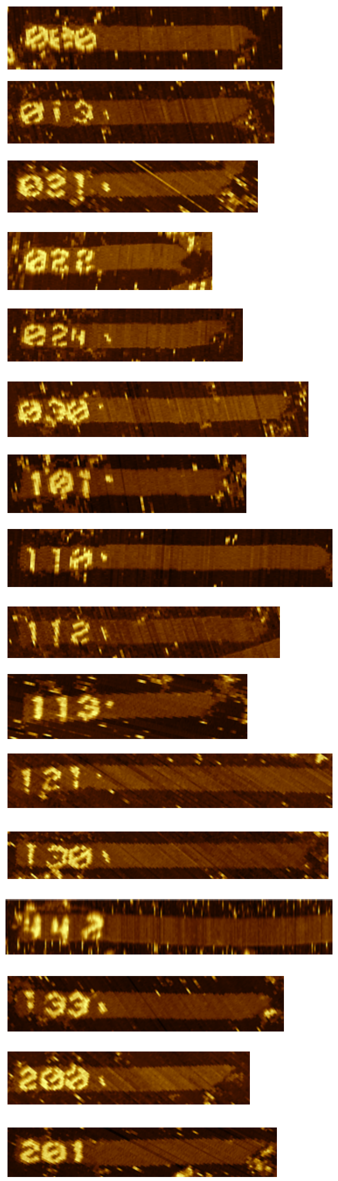




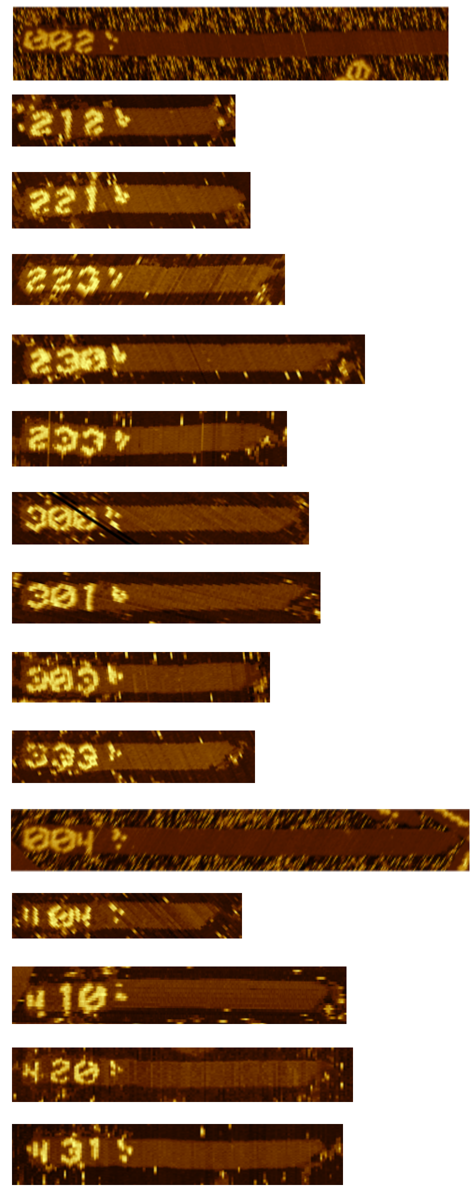
Testing of tile set: Parity on all 64 inputs

32 x No

$\sigma(000000) = 000$
$\sigma(000011) = 013$
$\sigma(000101) = 021$
$\sigma(000110) = 022$
$\sigma(001001) = 024$
$\sigma(001010) = 030$
$\sigma(001100) = 101$
$\sigma(001111) = 110$
$\sigma(010001) = 112$
$\sigma(010010) = 113$
$\sigma(010100) = 121$
$\sigma(010111) = 130$
$\sigma(011000) = 442$
$\sigma(011011) = 133$
$\sigma(011101) = 200$
$\sigma(011110) = 201$



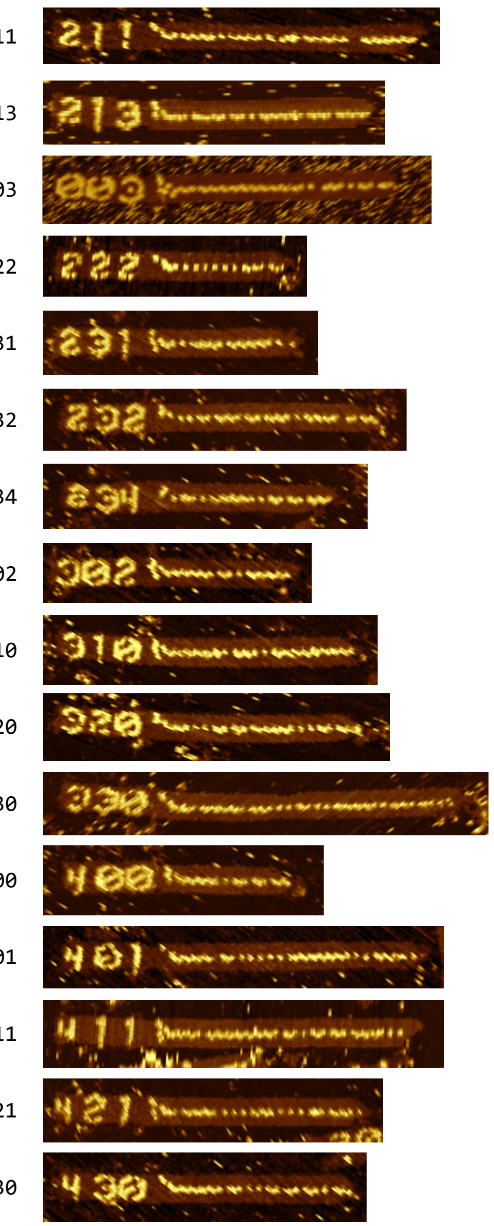
$\sigma(100001) = 002$
$\sigma(100010) = 212$
$\sigma(100100) = 221$
$\sigma(100111) = 223$
$\sigma(101000) = 230$
$\sigma(101011) = 233$
$\sigma(101101) = 300$
$\sigma(101110) = 301$
$\sigma(110000) = 303$
$\sigma(110011) = 333$
$\sigma(110101) = 004$
$\sigma(111001) = 404$
$\sigma(111010) = 410$
$\sigma(111100) = 420$
$\sigma(111111) = 431$



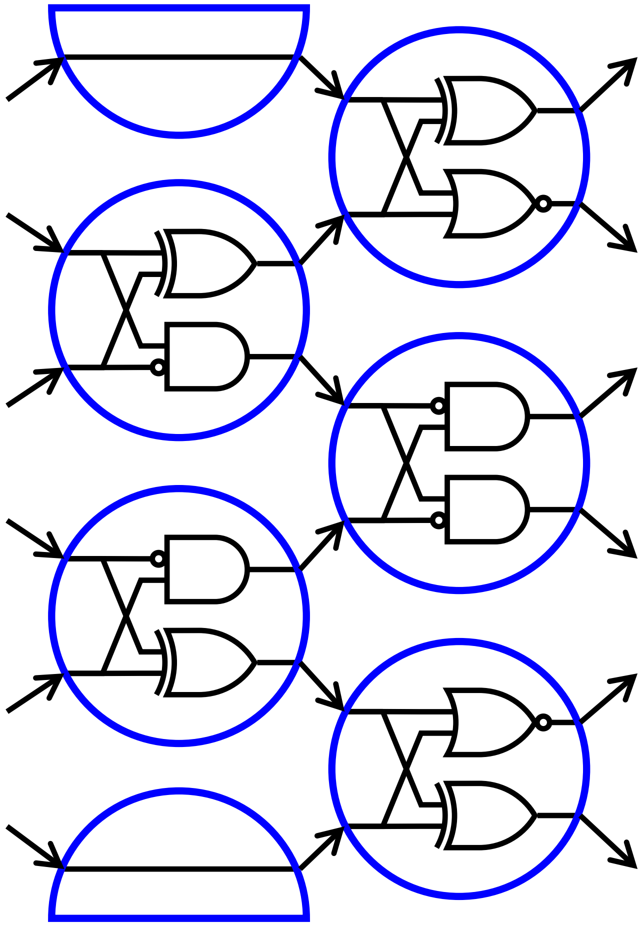
32 x Yes

$\sigma(000001) = 001$
$\sigma(000010) = 011$
$\sigma(000100) = 020$
$\sigma(000111) = 023$
$\sigma(001000) = 441$
$\sigma(001011) = 100$
$\sigma(001101) = 102$
$\sigma(001110) = 103$
$\sigma(010000) = 111$
$\sigma(010011) = 114$
$\sigma(010101) = 122$
$\sigma(010110) = 123$
$\sigma(011001) = 131$
$\sigma(011010) = 132$
$\sigma(011100) = 134$
$\sigma(011111) = 210$

$\sigma(100000) = 211$
$\sigma(100011) = 213$
$\sigma(100101) = 003$
$\sigma(100110) = 222$
$\sigma(101001) = 231$
$\sigma(101010) = 232$
$\sigma(101100) = 234$
$\sigma(101111) = 302$
$\sigma(110001) = 310$
$\sigma(110010) = 320$
$\sigma(110100) = 330$
$\sigma(110111) = 400$
$\sigma(111000) = 401$
$\sigma(111011) = 411$
$\sigma(111101) = 421$
$\sigma(111110) = 430$

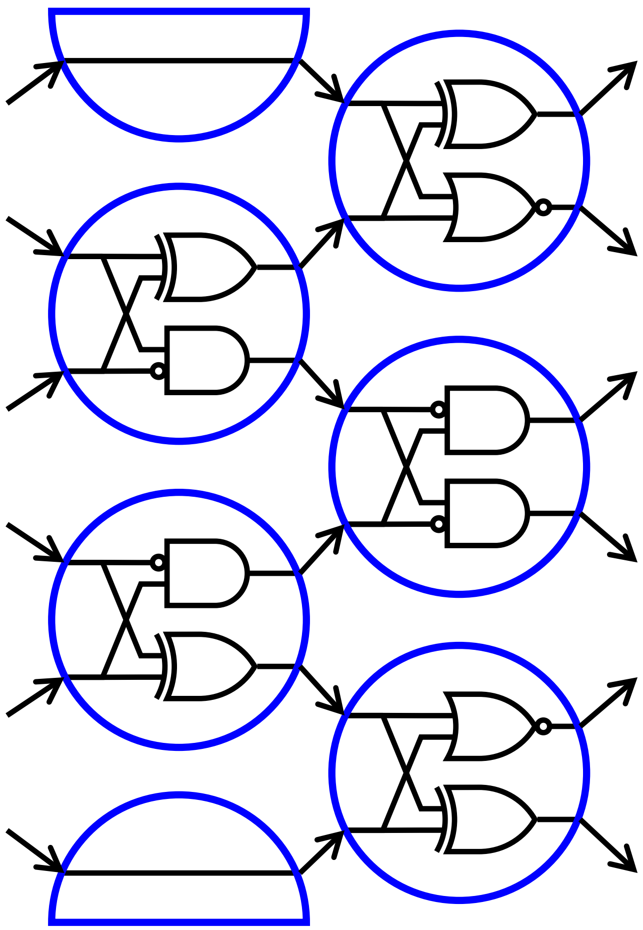


Is the input a multiple of 3?

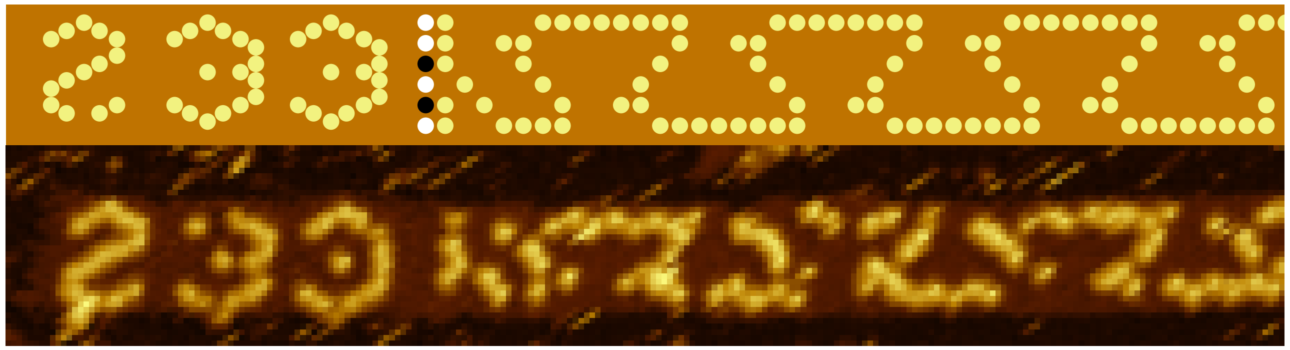


110101

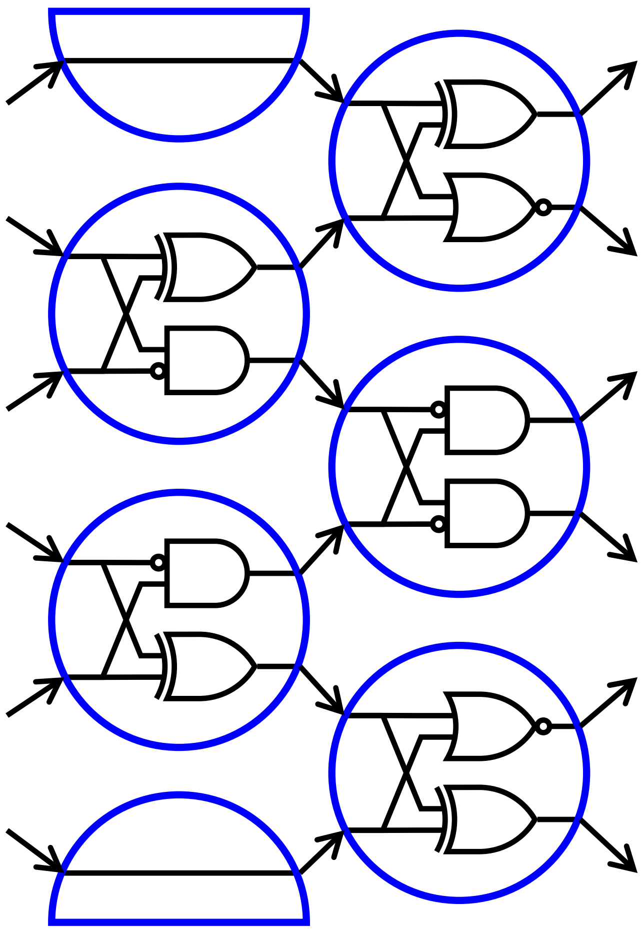
Is the input a multiple of 3?

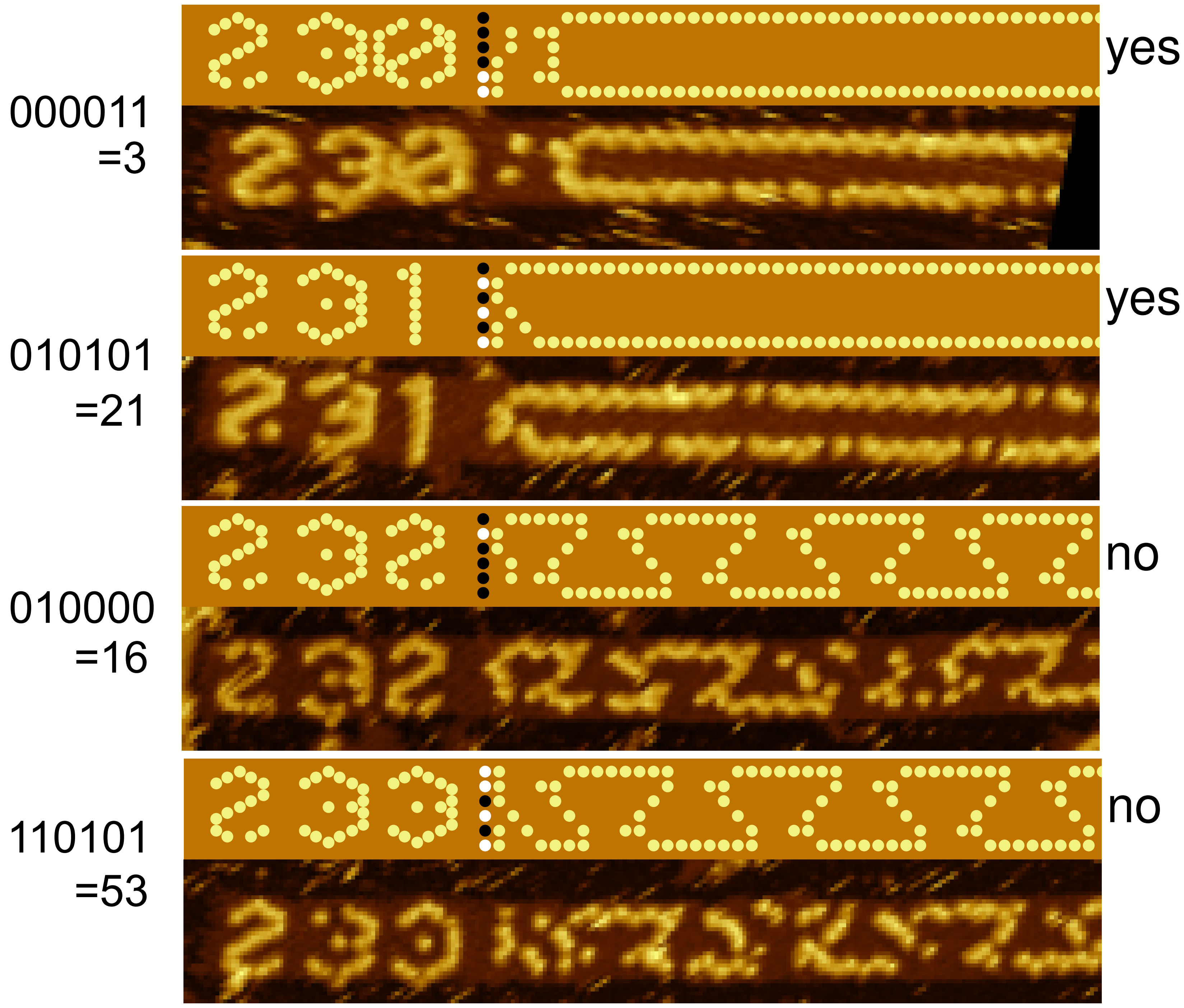


110101
=53

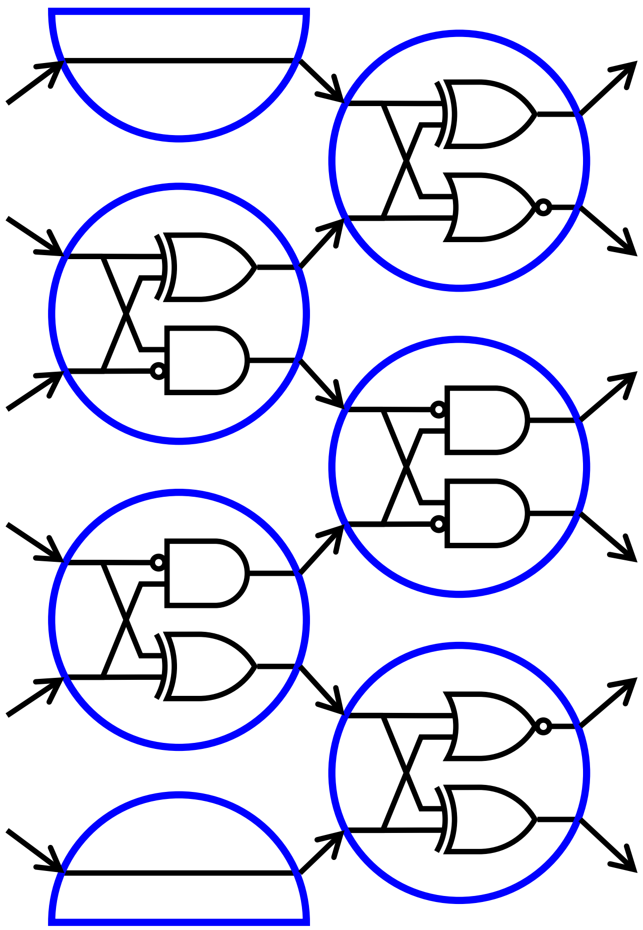


Is the input a multiple of 3?





Is the input a multiple of 3?



000011
=3

yes

010101
=21

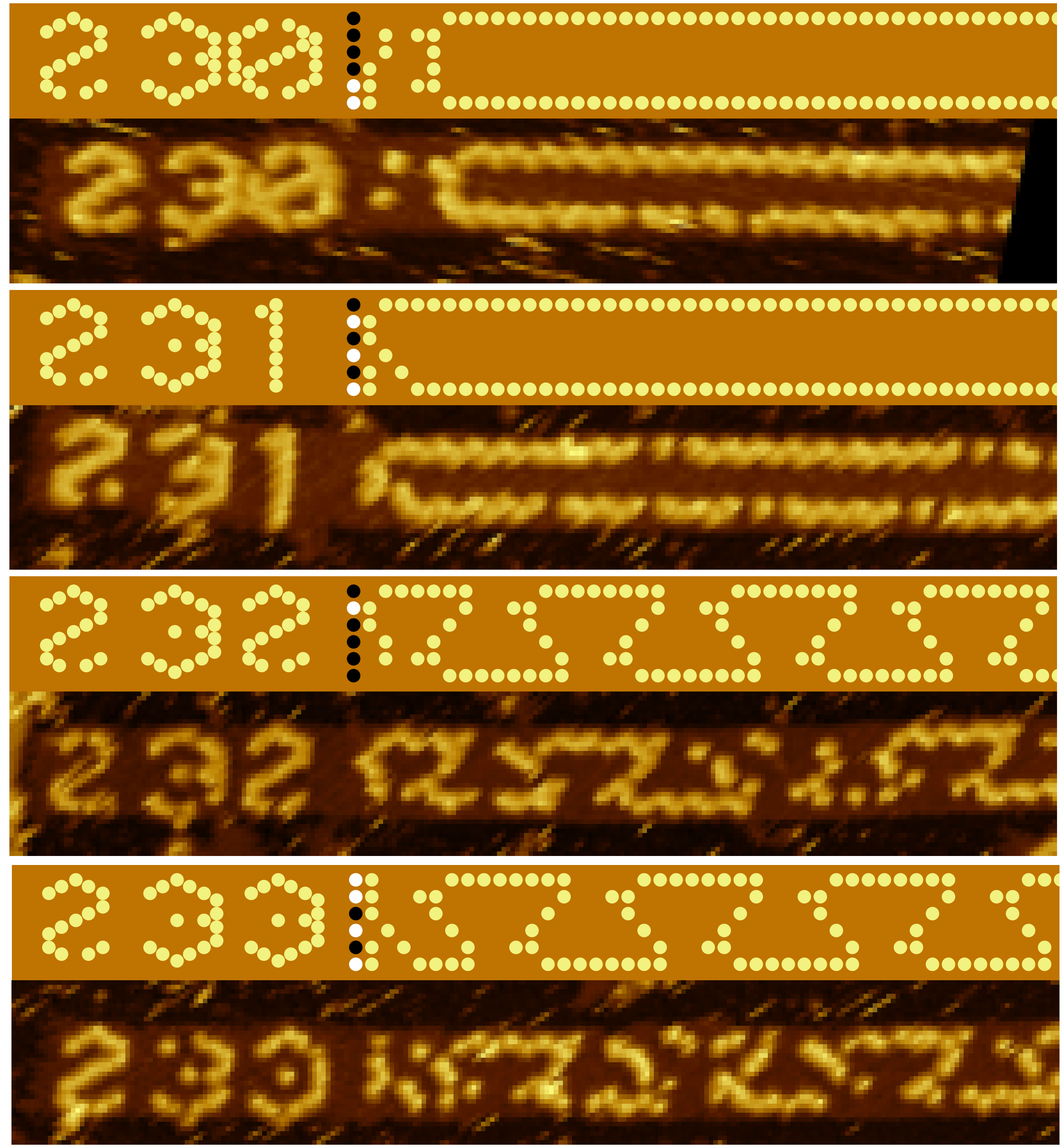
yes

010000
=16

no

110101
=53

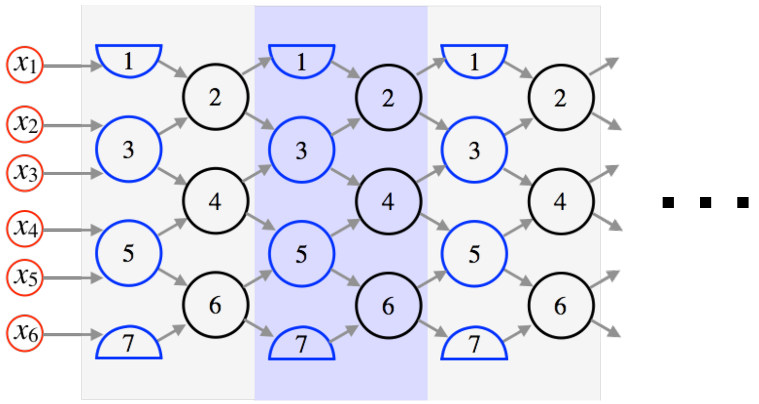
no

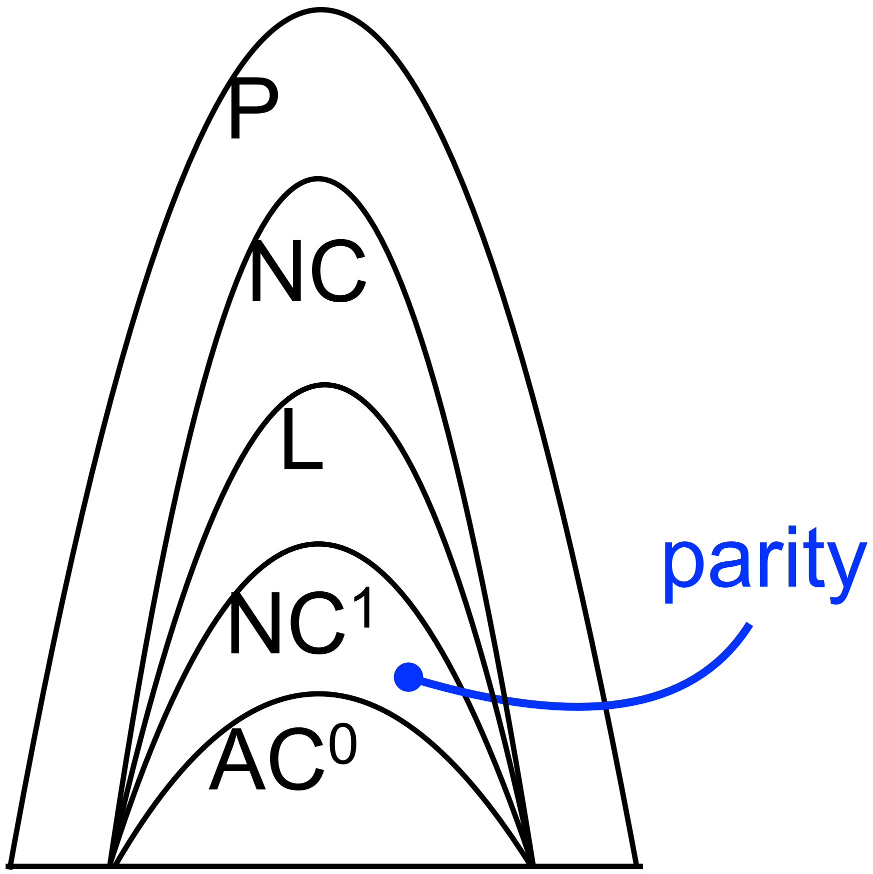


Erik Winfree

Computational power of this model?

The model is a rather restricted circuit model: “depth 2 layer”, restricted wiring within layer, repeated-layer, 0/1 signals on the wires. What can it compute?





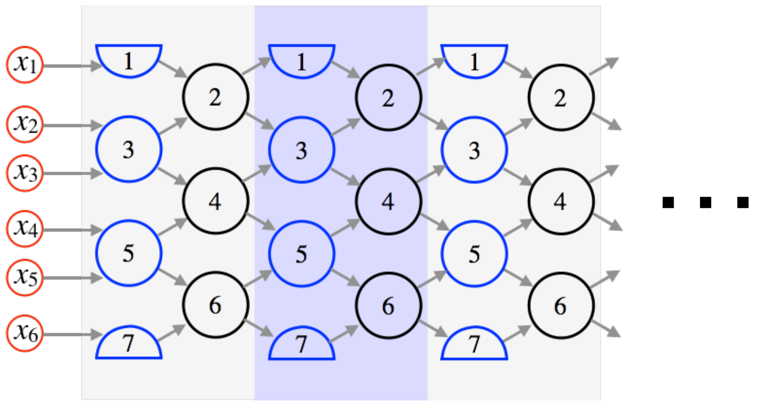
Something outside AC^0 (parity), no more than P (via explicit simulation for t layers)

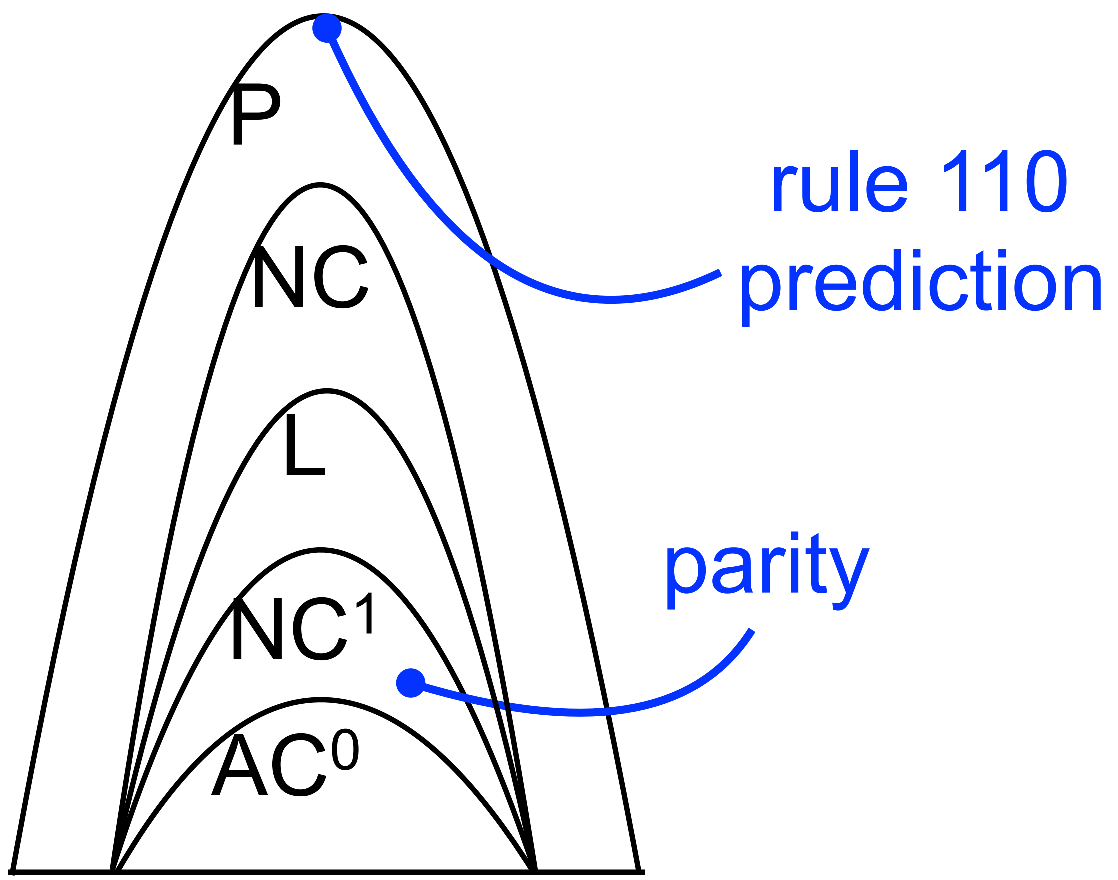
Classes of problems:

- AC^0: constant depth, poly size, Boolean circuits with arbitrary fanin gates
- NC^1: log depth, poly size, Boolean circuits with fanin ≤ 2 gates
- L: deterministic log space Turing machines
- P: deterministic polynomial time Turing machines

Computational power of this model?

The model is a rather restricted circuit model: “depth 2 layer”, restricted wiring within layer, repeated-layer, 0/1 signals on the wires. What can it compute?



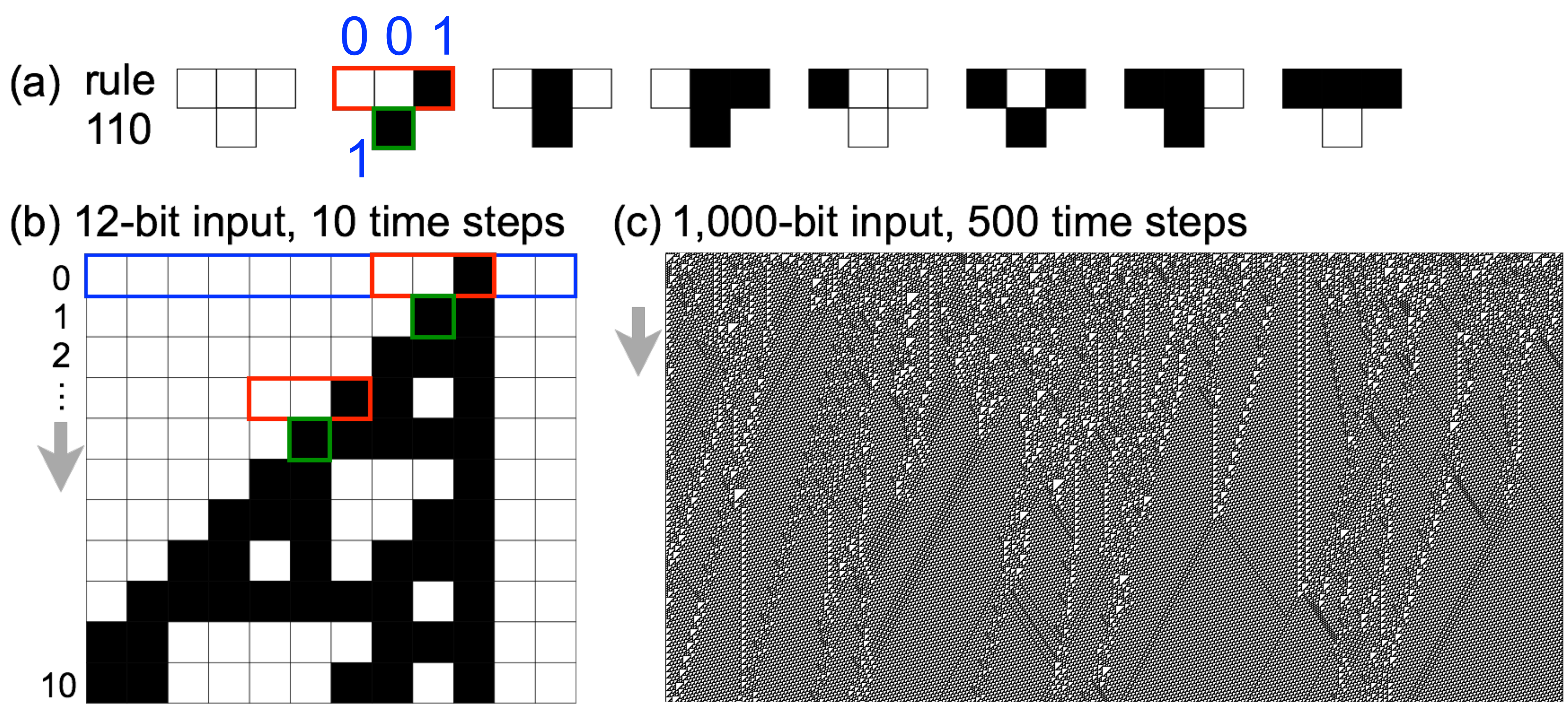


Something outside AC^0 (parity), no more than P (via explicit simulation for *t* layers)

Classes of problems:

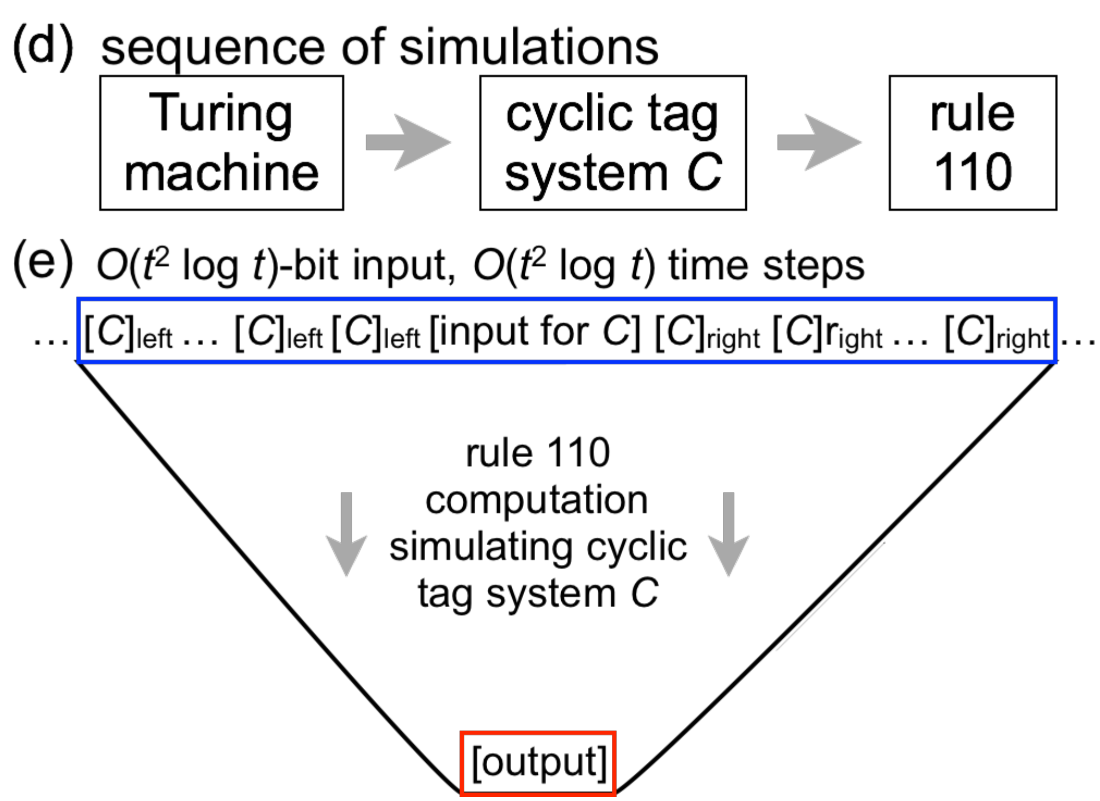
- AC^0: constant depth, poly size, Boolean circuits with arbitrary fanin gates
- NC^1: log depth, poly size, Boolean circuits with fanin ≤ 2 gates
- L: deterministic log space Turing machines
- P: deterministic polynomial time Turing machines

Rule 110



Theorem: Let *M* be a Turing machine that runs in time *t*, rule 110 simulates *M* in $O(t^2 \log t)$ steps

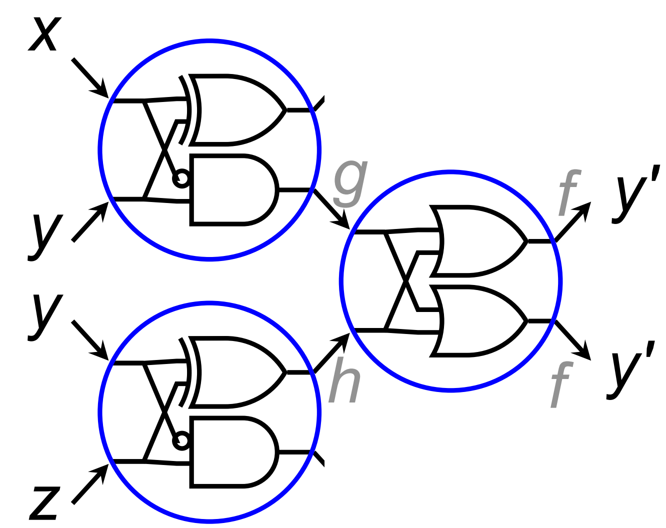
[Cook 2004]
[Neary, Woods, 2006]
[Neary, PhD thesis]

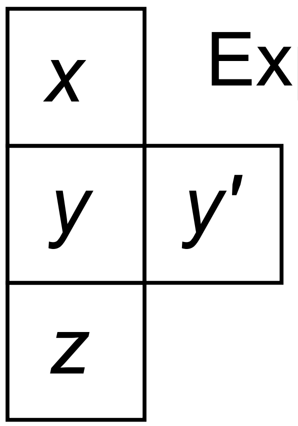


Simulation of rule 110

x y z	*x y z*	
$F(0,0,0)=0$	$F(1,0,0)=0$	
$F(0,0,1)=1$	$F(1,0,1)=1$	rule 110
$F(0,1,0)=1$	$F(1,1,0)=1$	truth table
$F(0,1,1)=1$	$F(1,1,1)=0$	

IBC sub-circuit





Express the rule as *f*(*g*(*x*,*y*),*h*(*y*,*z*))

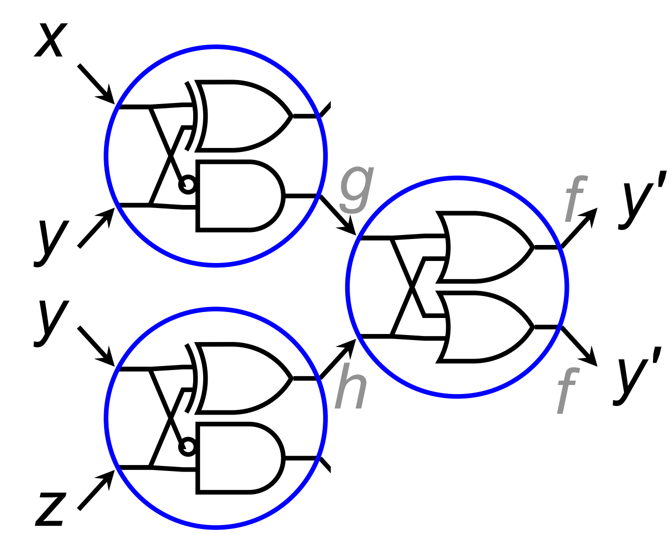
$$y' = (\neg x \wedge y) \vee (y \otimes z)$$

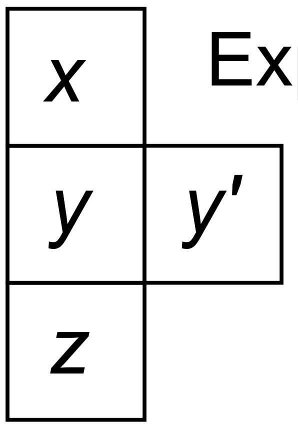
Simulation of rule 110

$x\,y\,z$	$x\,y\,z$
$F(0,0,0)=0$	$F(1,0,0)=0$
$F(0,0,1)=1$	$F(1,0,1)=1$
$F(0,1,0)=1$	$F(1,1,0)=1$
$F(0,1,1)=1$	$F(1,1,1)=0$

rule 110 truth table

IBC sub-circuit

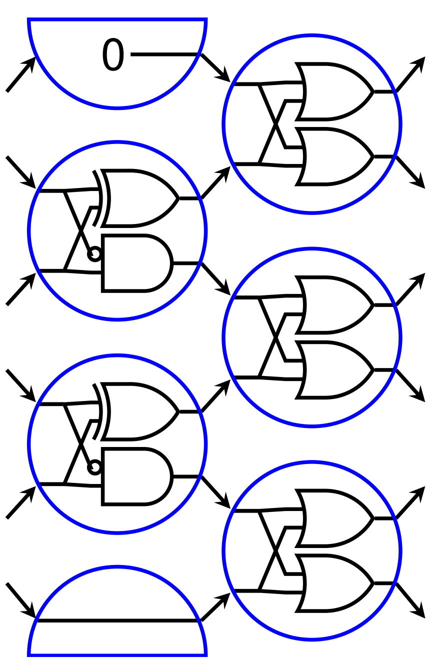




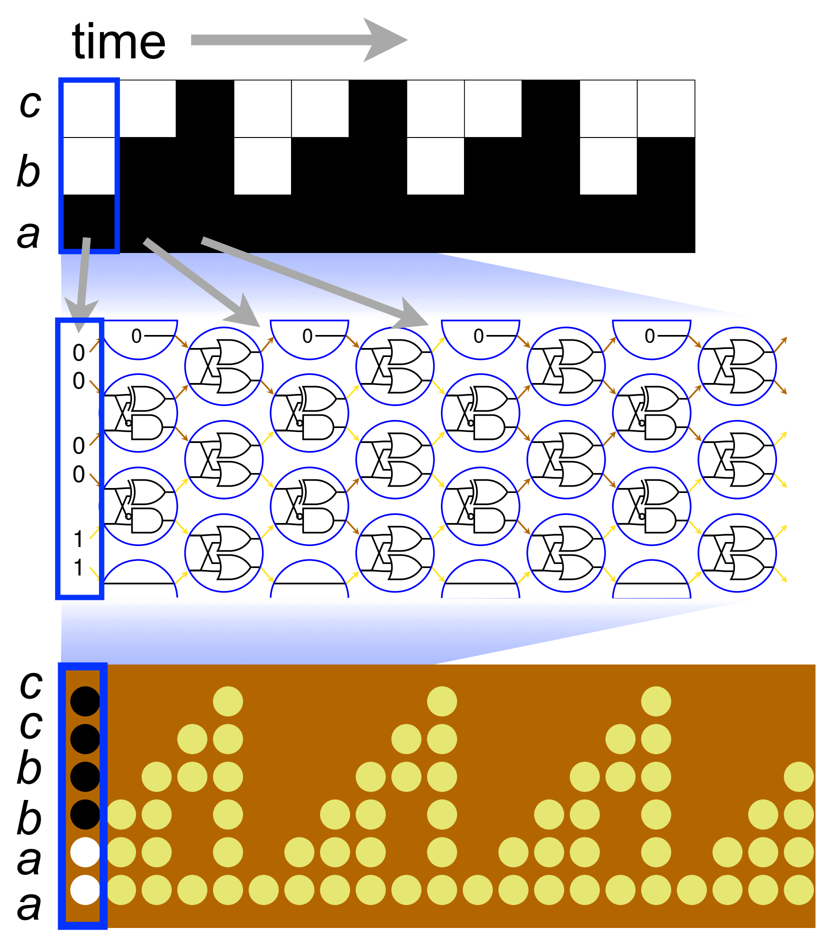
Express the rule as *f*(*g*(*x*,*y*),*h*(*y*,*z*))

$$y' = (\neg x \wedge y) \vee (y \otimes z)$$

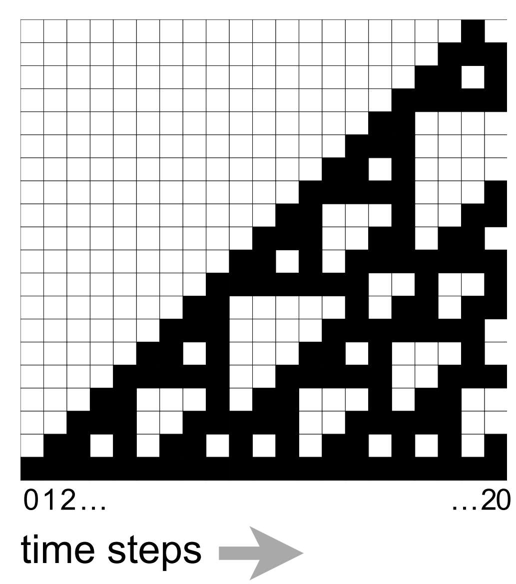
6-bit IBC to simulate 3 bits of rule 110



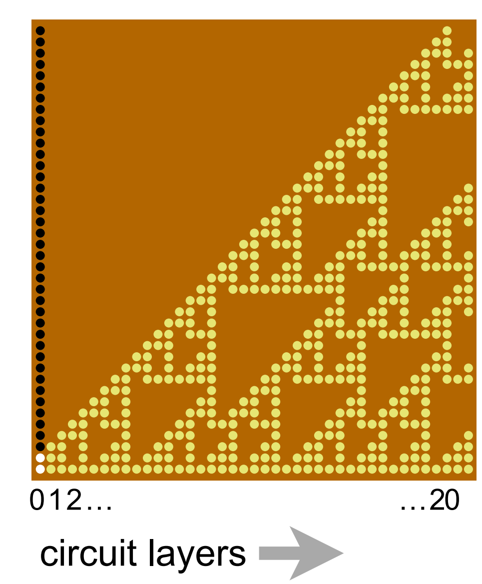
IBC simulation on 3 bits



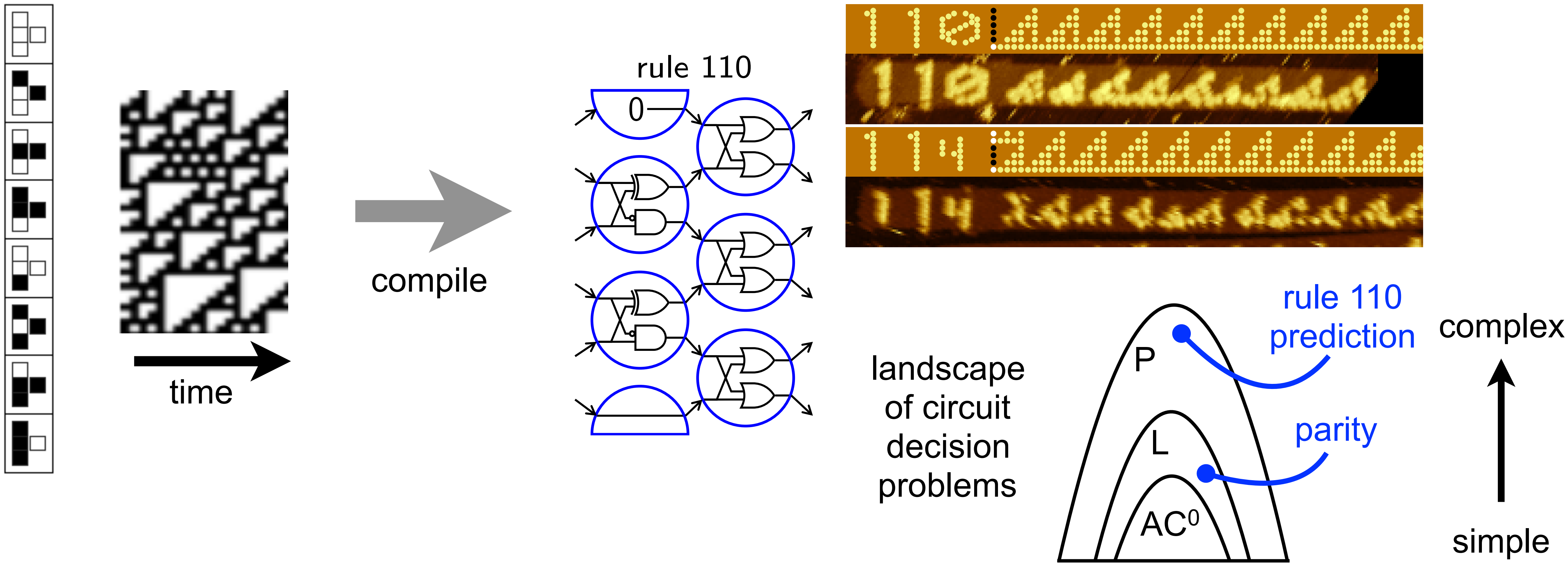
rule 110



IBC



RULE110 circuit: simulation of cellular automata

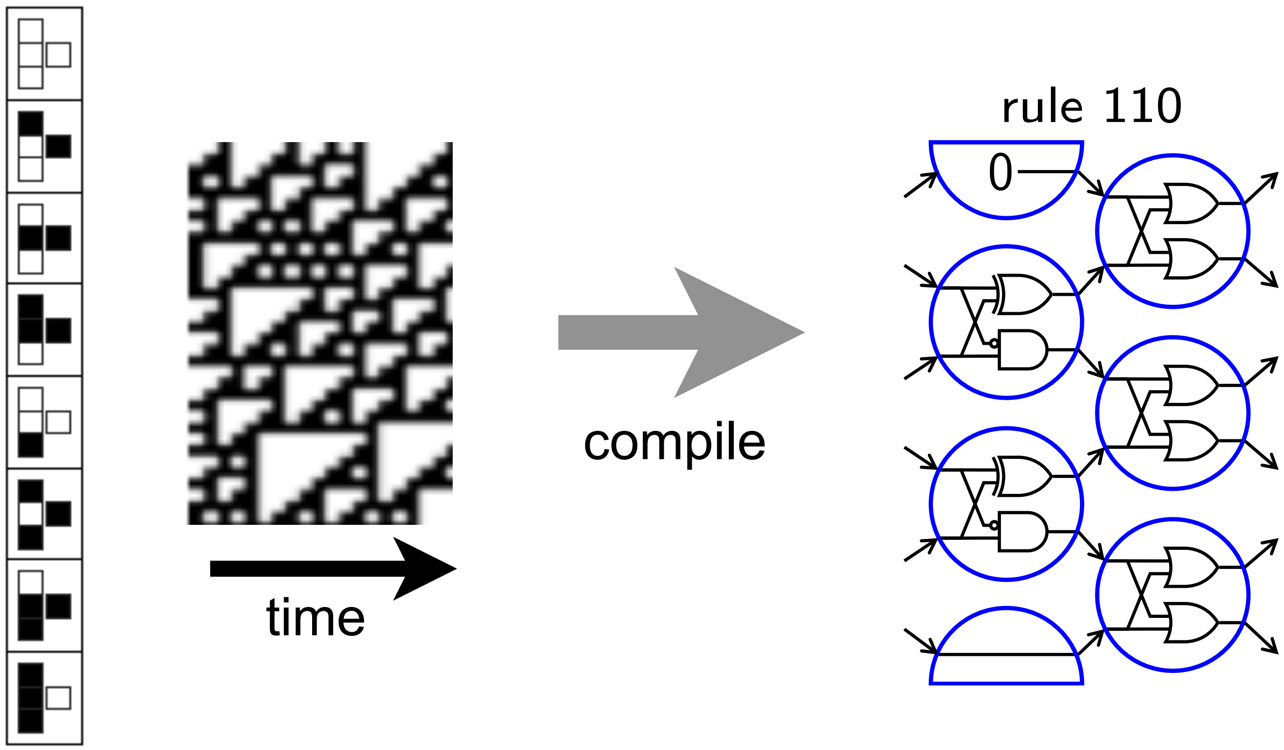


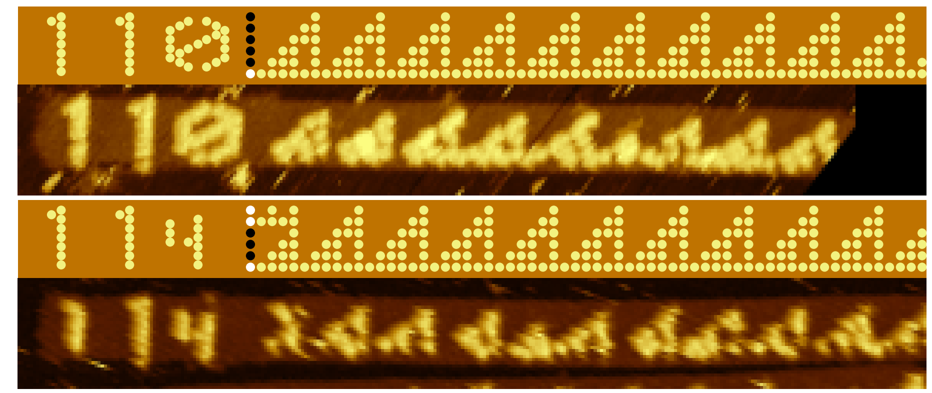
Theorem: Let *M* be a single-tape Turing machine that runs in time *t*, then $O(t^2 \log t)$-bit 1-layer circuits (IBCs) simulate *M*

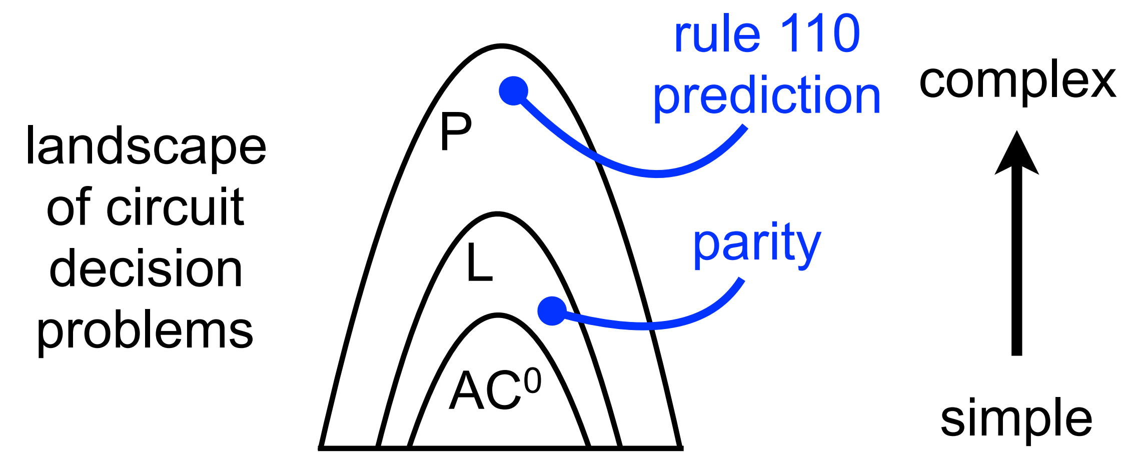
IBCs efficiently simulate any algorithm

[Cook 2004]
[Neary, Woods, 2006]
[Neary, PhD thesis]

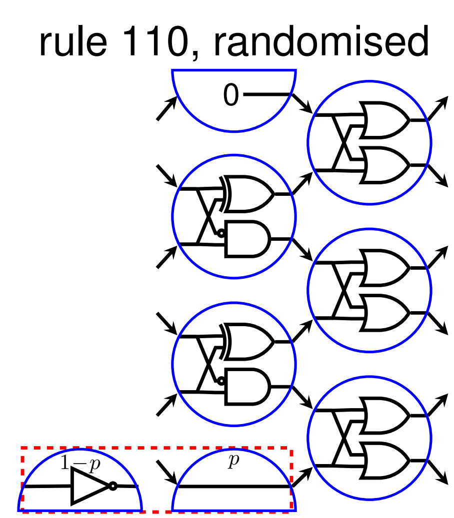
RULE110 circuit: simulation of cellular automata







Theorem: Let M be a single-tape Turing machine that runs in time t, then $O(t^2 \log t)$-bit 1-layer circuits (IBCs) simulate M

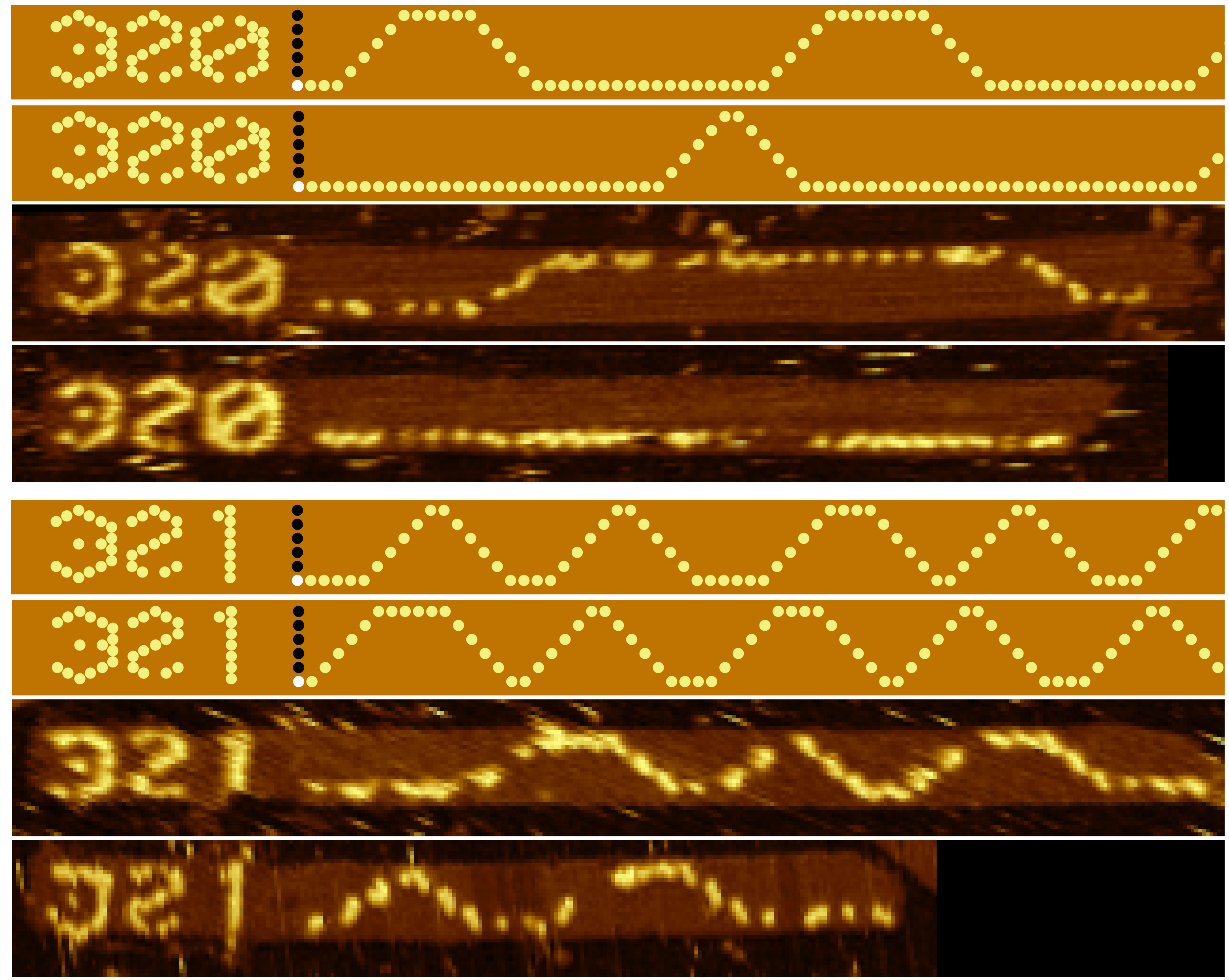


IBCs efficiently simulate any algorithm

[Cook 2004]
[Neary, Woods, 2006]
[Neary, PhD thesis]

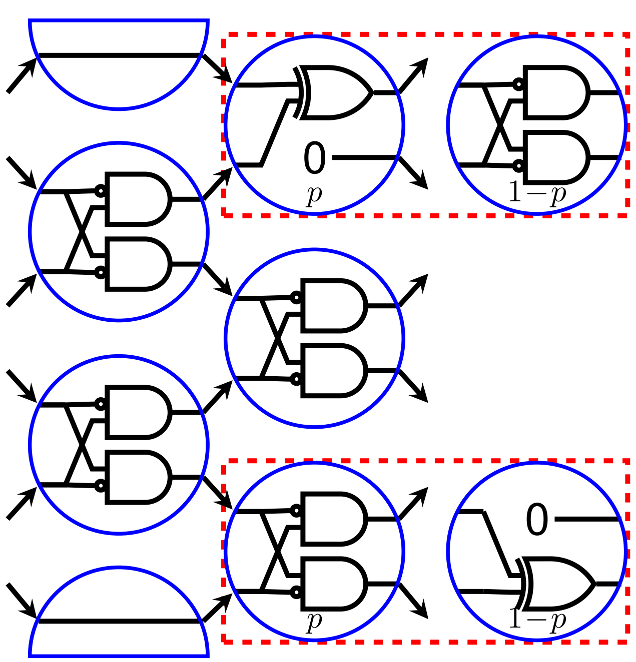
Open: characterise power of randomised model

California surf: WAVES

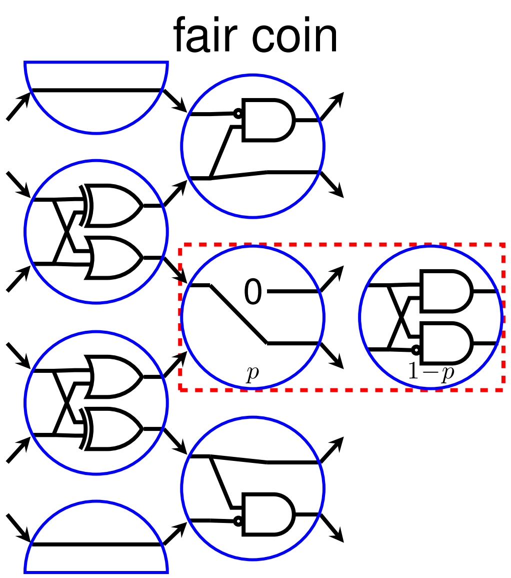


Pr[create wave]=0.1
Pr[crash wave] =0.5

Pr[create wave]=0.5
Pr[crash wave] =0.5



FAIRCOIN: Unbiased bit from biased coin

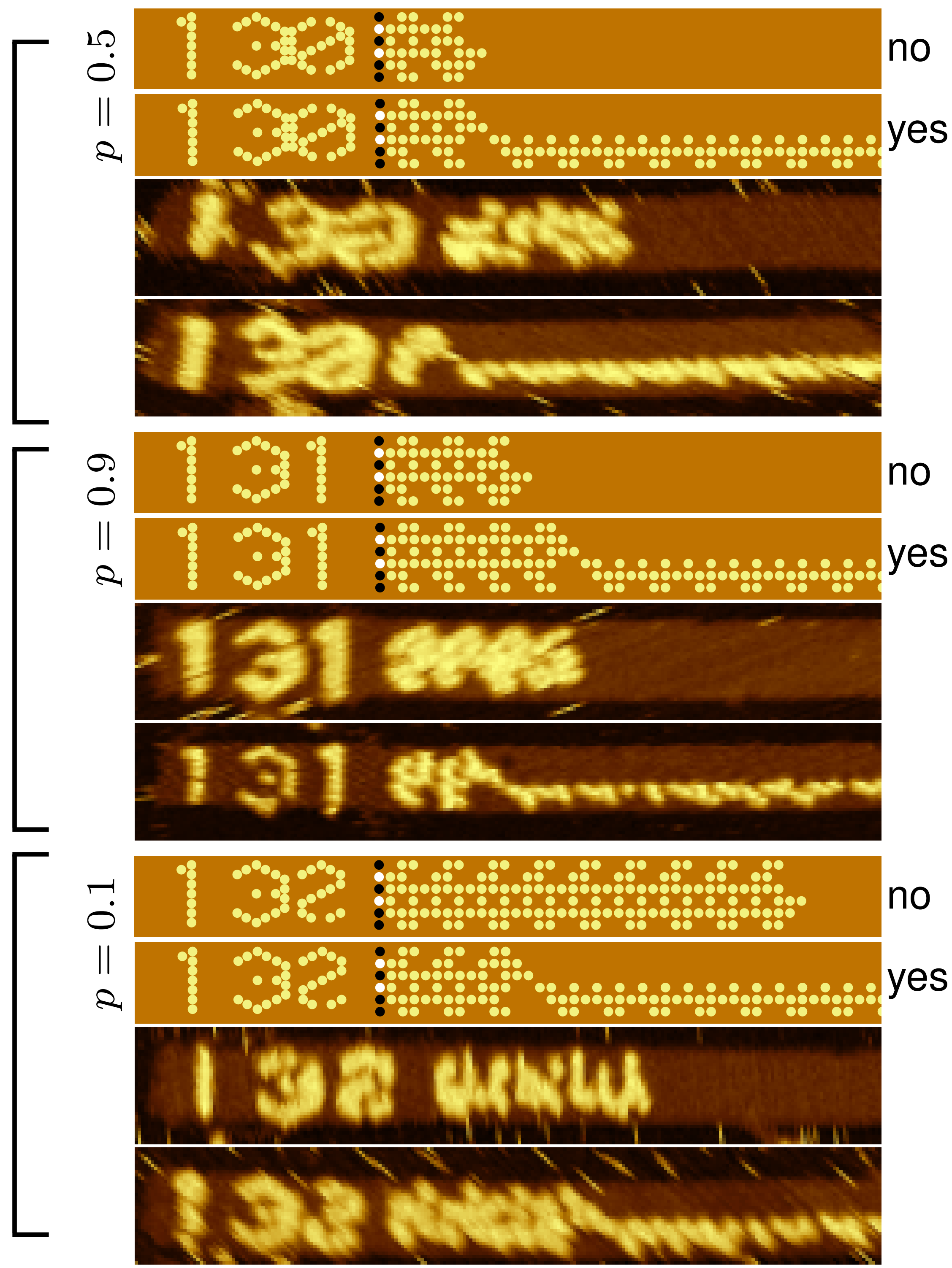


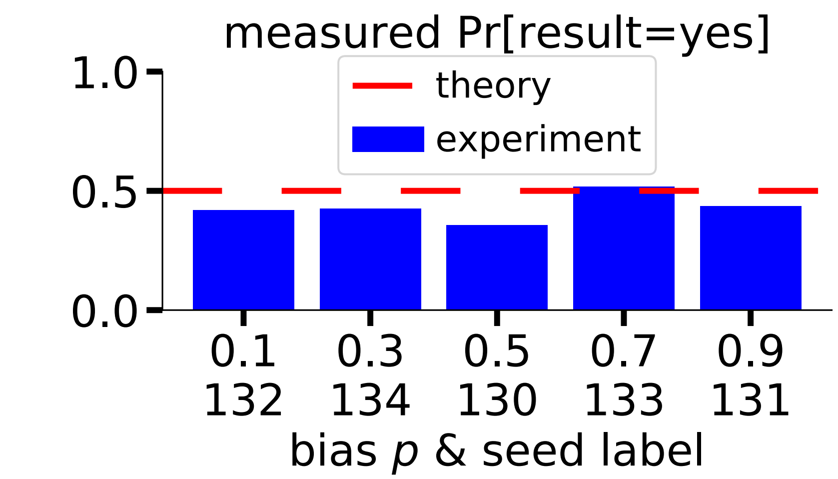
1:1

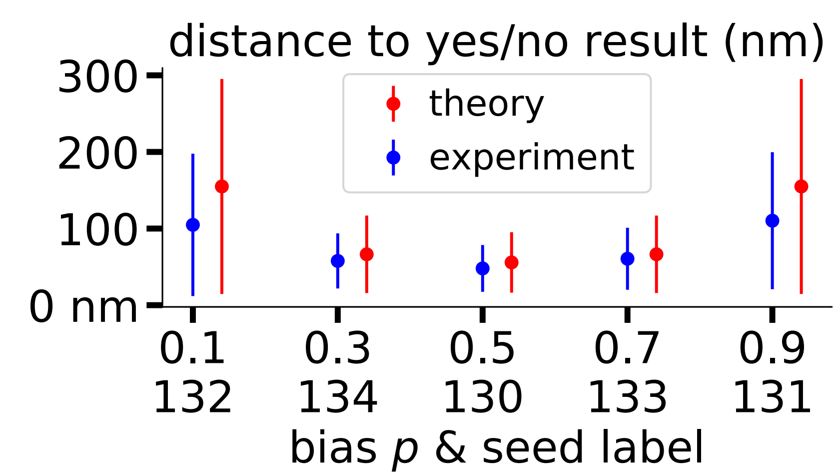
9:1

1:9

heads:tails



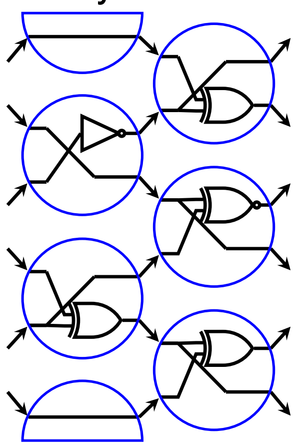




Dave Doty

Counting to 63

cycle 63



Circuit with 63 distinct strings

1 2 3 …

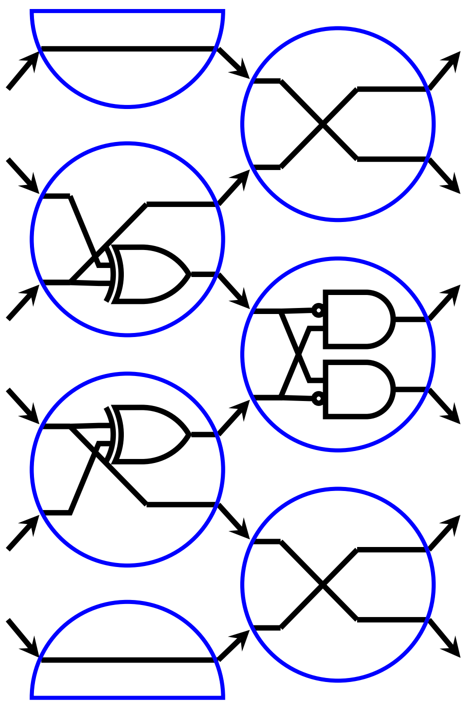
…62 63 1 2 …

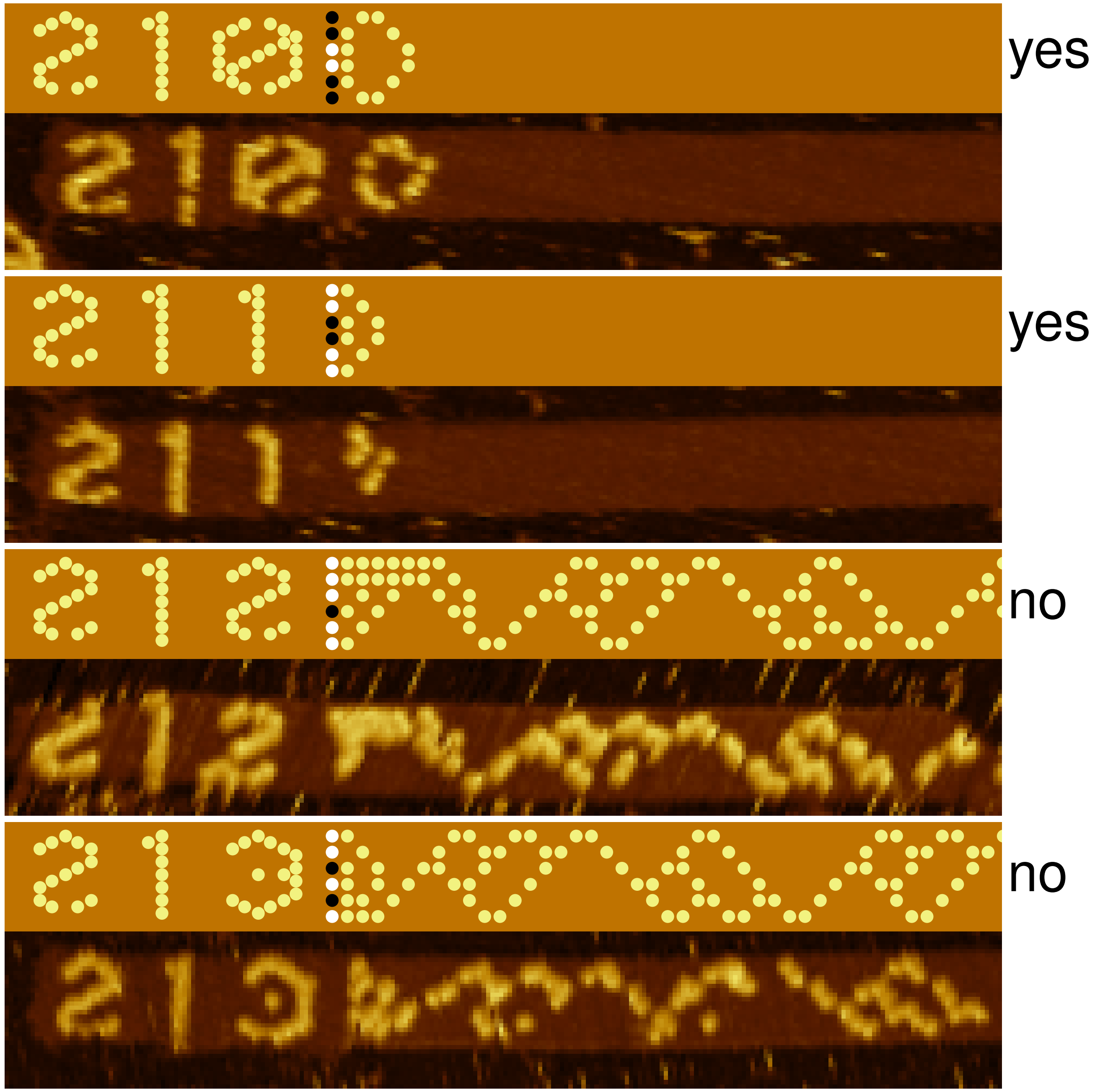
42

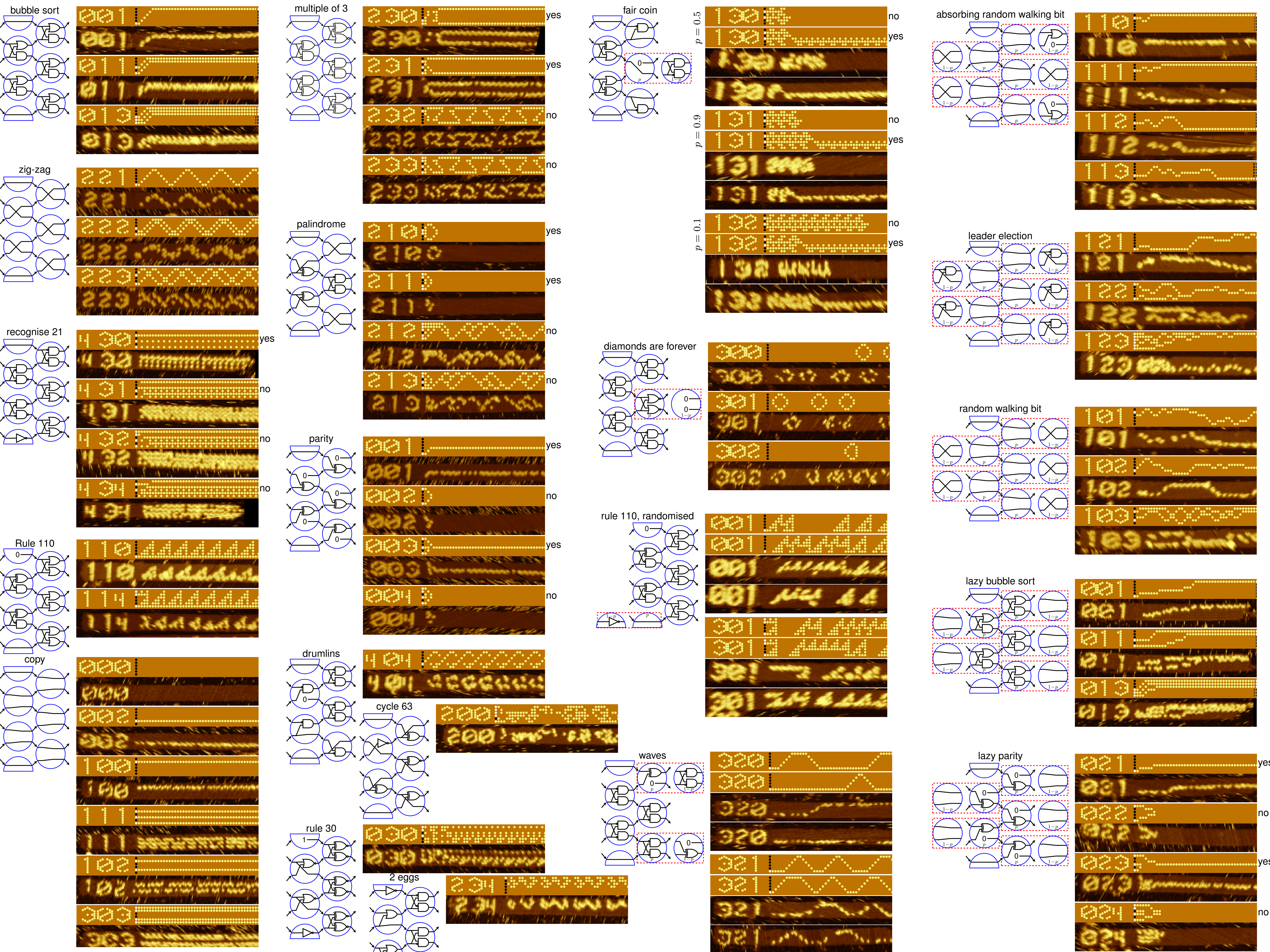
Is there a 64-counter?

No!
Proof by Tristan Stérin, Maynooth University

Palindromes: high communication complexity



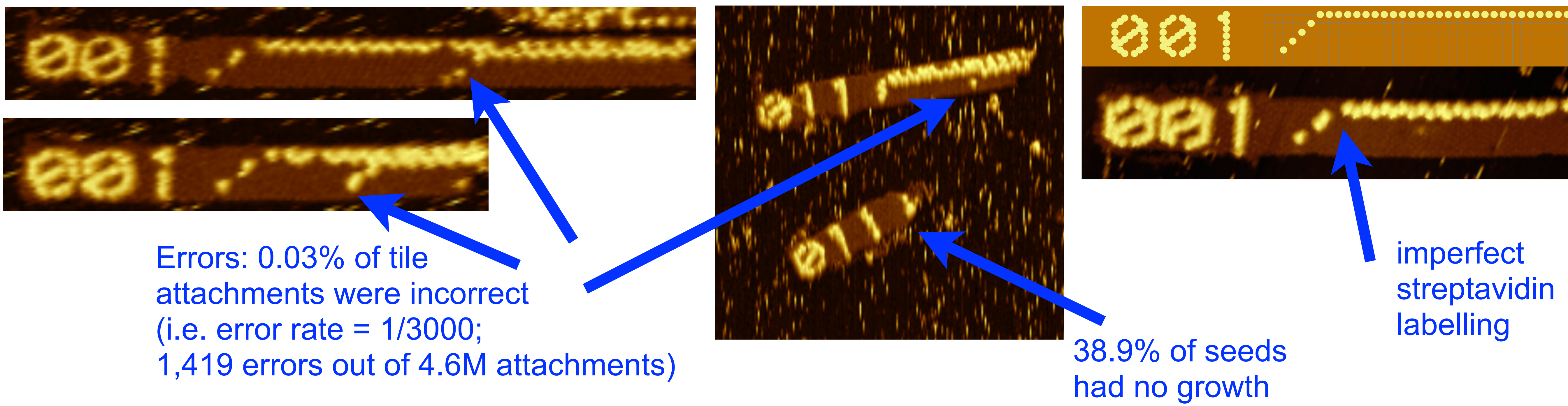



bubble sort
zig-zag
recognise 21
yes
no
no
no
Rule 110
copy
multiple of 3
yes
yes
no
no
palindrome
yes
yes
no
no
parity
yes
no
yes
no
drumlins
cycle 63
rule 30
2 eggs
fair coin
$p = 0.5$
no
yes
$p = 0.9$
no
yes
$p = 0.1$
no
yes
diamonds are forever
rule 110, randomised
waves
absorbing random walking bit
leader election
random walking bit
lazy bubble sort
lazy parity
yes
no
yes
no

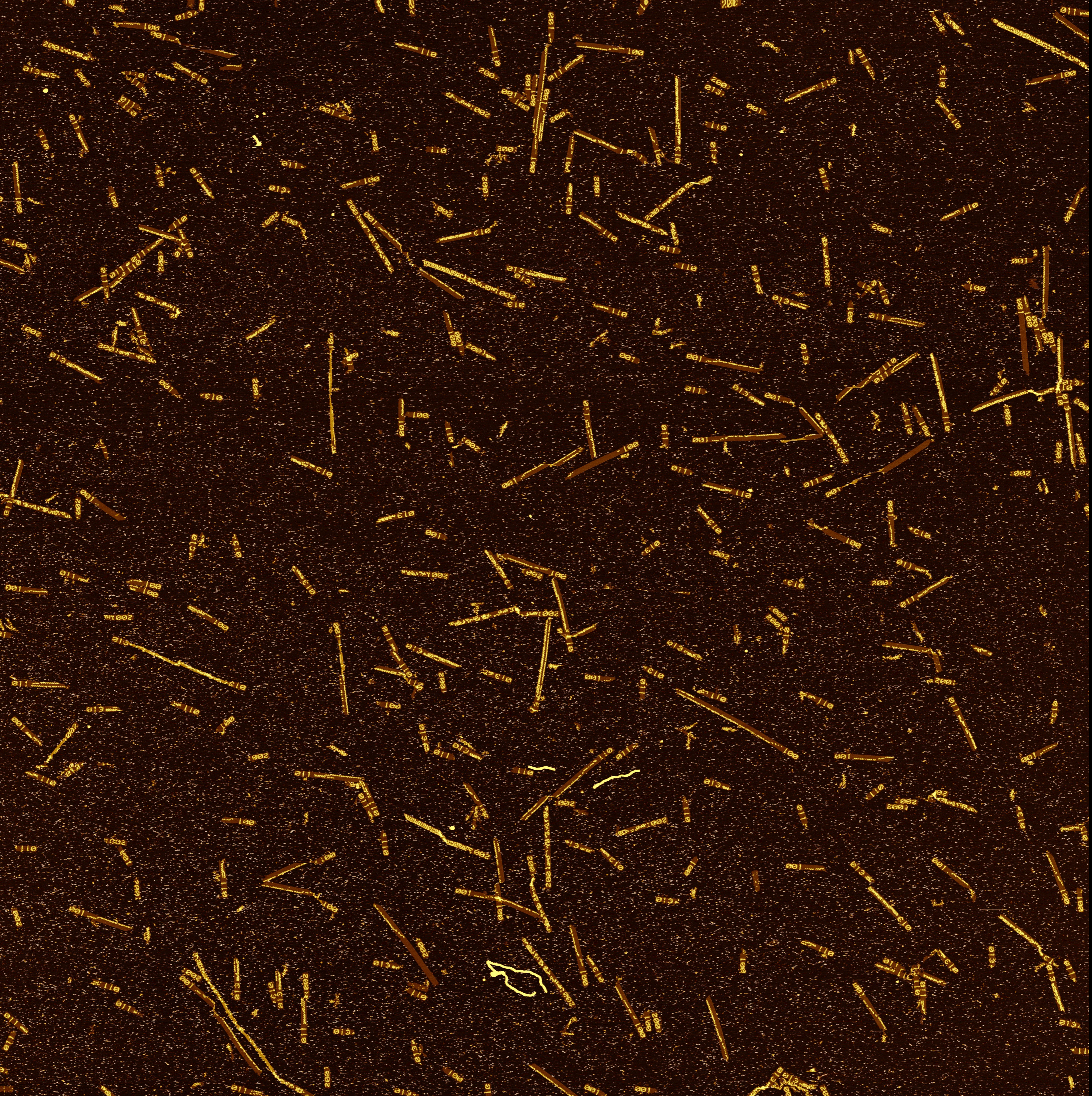
How well did the 21 circuits work?

Extensive testing of all 355 tiles:
- **every tile type** was used in some circuit
- for many circuits **tested all tile types for that circuit**
- ran one circuit on **all 64 inputs**

Analysed ~12k nanotubes with ~5M tile attachments:

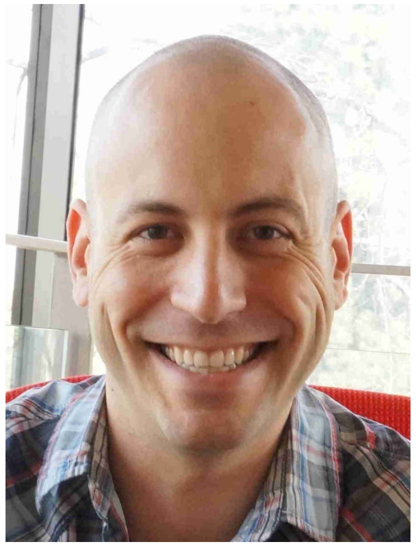


Reprogrammable: demonstrated many new self-assembly programs
Scaling up: 15x more tile types than previous algorithmic self-assembly systems
Low error: Careful sequence design; Proofreading
Good structure: Nanotube lattice & hardcoded rows
Lots of tile types: Long SST domains



raw data
8μm x 8μm

Acknowledgements

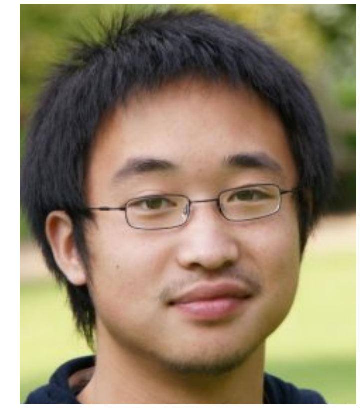


Dave Doty
UC Davis

Erik Winfree
Caltech

C Myhrvold
Harvard

Joy Hui
Harvard



Felix Zhou
Oxford

Peng Yin
Harvard

Caltech

Inria

UC Davis

Harvard

Maynooth University
National University of Ireland Maynooth

Woods, Doty, Myhrvold, Hui, Zhou, Yin, Winfree. *Diverse and robust molecular algorithms using reprogrammable DNA self-assembly.* Nature. 567 (7748), 366 2019

Special thanks to: Constantine Evans, Ashwin Gopinath, Paul Rothemund, Sungwook Woo, Cody Geary, Cris Moore, Chris Thachuk, Rizal Hariadi, Rebecca Schulman

NASA NSF

- 1219274 Doty, Woods
- 1162589 Winfree Yin Doty Woods
- 0832824 & 1317694 Molecular Programming Project. Winfree et al.
- NASA #NNX13AJ56G. Woods.

Fin?

Ongoing and future work:

- Self-assembly
- Molecular robotics
- Self-replication

We are looking for postdocs and PhD students!

damien.woods@mu.ie