Crystals that think
about how they're growing

David Doty

joint work with Damien Woods, Erik Winfree, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin

University of New Mexico, Computer Science Colloquium, Sept 10, 2019

Caltech   Inria Paris   UC Davis   Harvard

Acknowledgements

Caltech

Inria Paris

UC Davis

Harvard

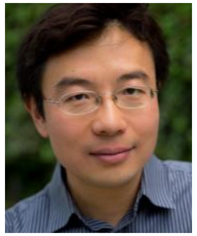
Damien Woods
(co-first author)

Erik Winfree
(PI)

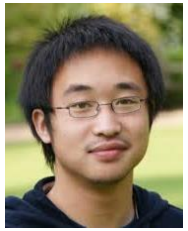
co-authors

Cameron Myhrvold

Peng Yin



Felix Zhou



Joy Hui

lab/science help

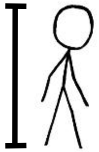
Sungwook Woo
Constantine Evans
Sarina Mohanty
Niranjan Srinivas
Deborah Fygenson
Yannick Rondolez
Mingjie Dai
Nikhil Gopalkrishnan
Chris Thachuk
Nadine Dabby
Jongmin Kim
Paul Rothemund
Bryan Wei
Cody Geary
Ashwin Gopinath

Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. Damien Woods†, David Doty†, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, Erik Winfree. Nature 2019. †*These authors contributed equally*.

Ljubljana Marshes Wheel. 5k years old

Building things

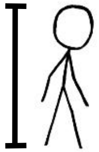
Newgrange, Ireland. 5.2k years old

Building things by hand: use tools! Great for scale of $10^{\pm 2} \times$ 

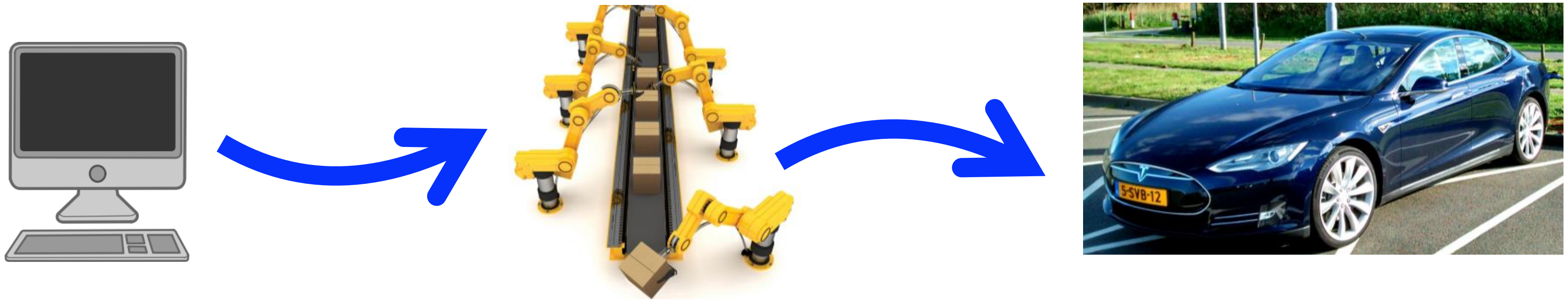
Ljubljana Marshes Wheel. 5k years old

Building things

Newgrange, Ireland. 5.2k years old

Building things by hand: use tools! Great for scale of $10^{\pm 2} \times$ 

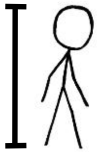
Building tools that build things: specify target object with a computer program that then controls the manufacturing process



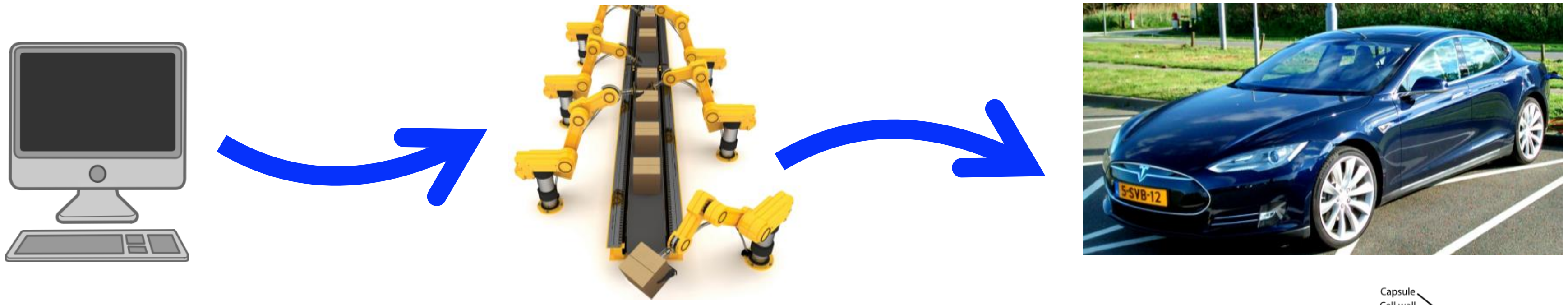
Ljubljana Marshes Wheel. 5k years old

Building things

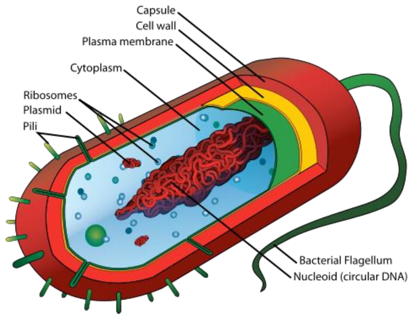
Newgrange, Ireland. 5.2k years old

Building things by hand: use tools! Great for scale of $10^{\pm 2}$ × 

Building tools that build things: specify target object with a computer program that then controls the manufacturing process

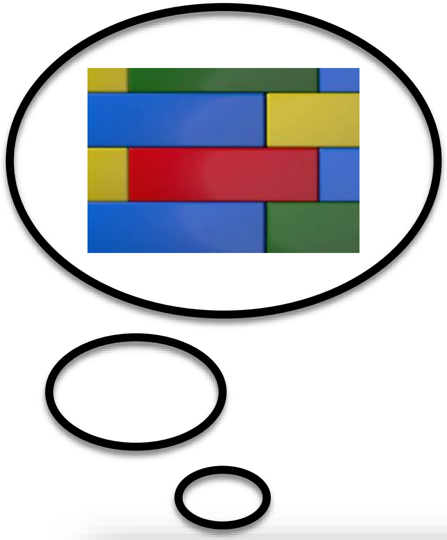


Programming **things to build themselves**: for building in small wet places where our hands or tools can't reach

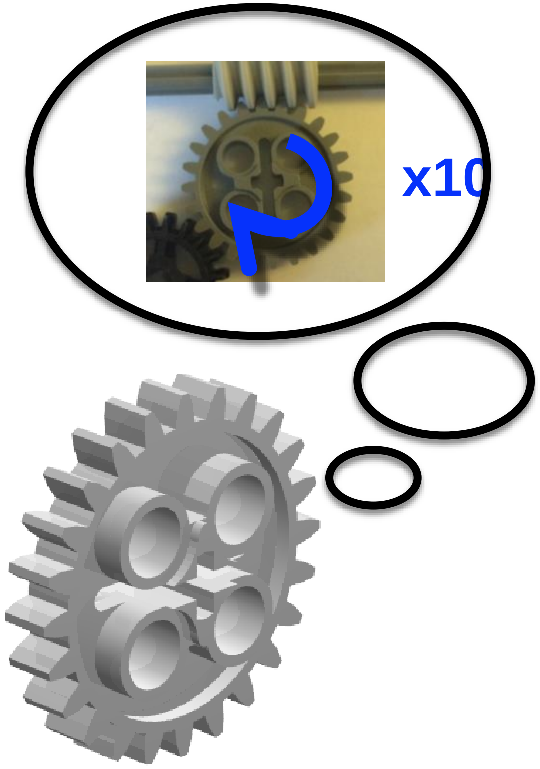


Mariana Ruiz Villarreal

Things that build themselves

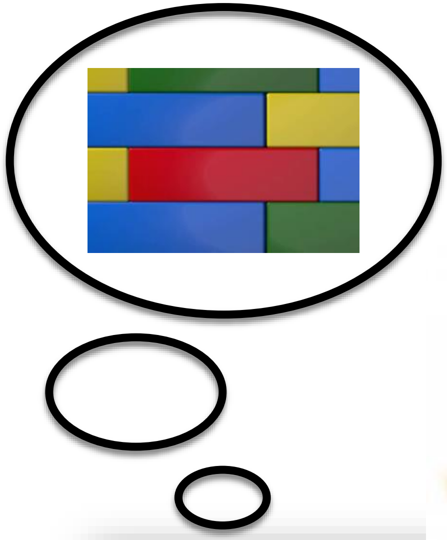


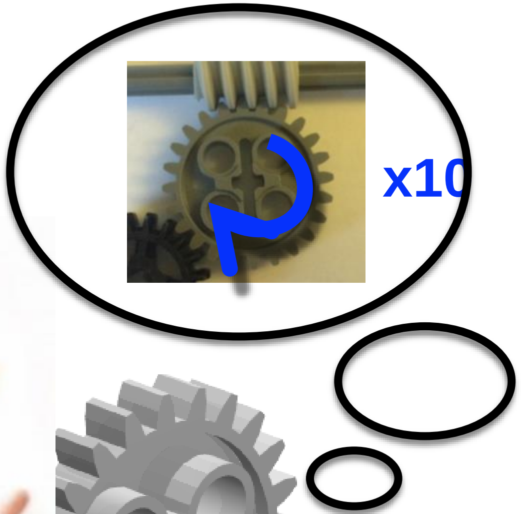
I want to stick below blue & yellow and above blue & green

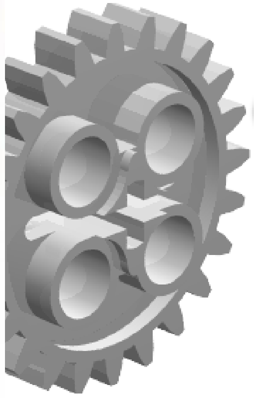


Our topic: self-assembling molecules that compute as they build themselves

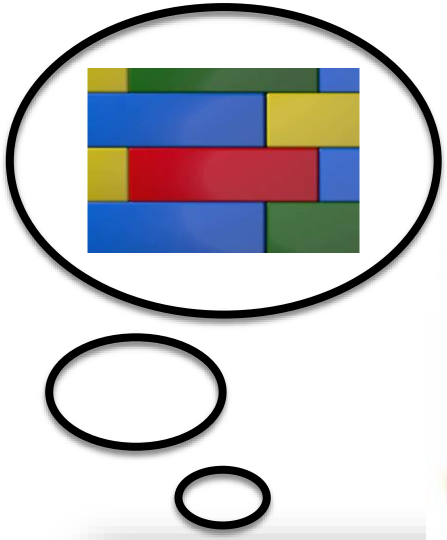
Things that build themselves



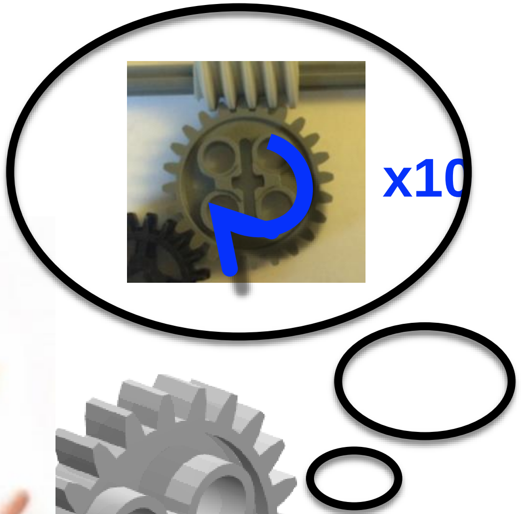




Our topic: self-assembling molecules that compute as they build themselves



Things that build themselves



Our topic: self-assembling molecules that compute as they build themselves

Hierarchy of abstractions

Bits:	**Boolean circuits compute**
Tiles:	Tile self-assembly implements circuits
DNA:	DNA strands implement tiles

Harmonious arrangement

0

1

1

0

1

1

Harmonious arrangement

0

1

1

0

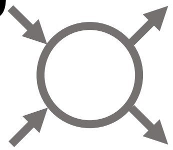
1

1

Harmonious arrangement

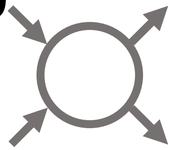
0	1
1	1
1	1
0	1
1	0
1	0

Harmonious arrangement



	0		1	
	1		1	
	1		1	
	0		1	
	1		0	
	1		0	

Harmonious arrangement

0 1

1 0

1

0

1

1

1

1

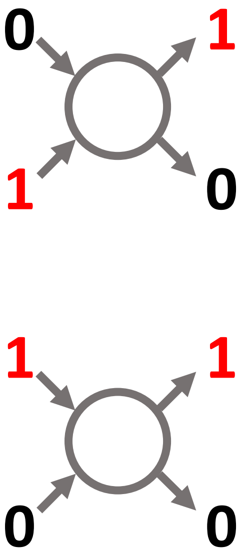
1

1

0

0

Harmonious arrangement



1

1

1

1

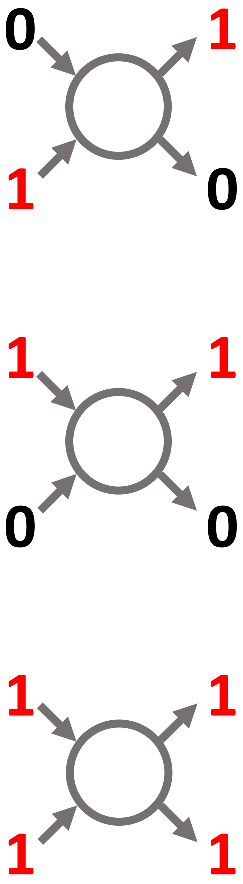
1

1

0

0

Harmonious arrangement



1

1

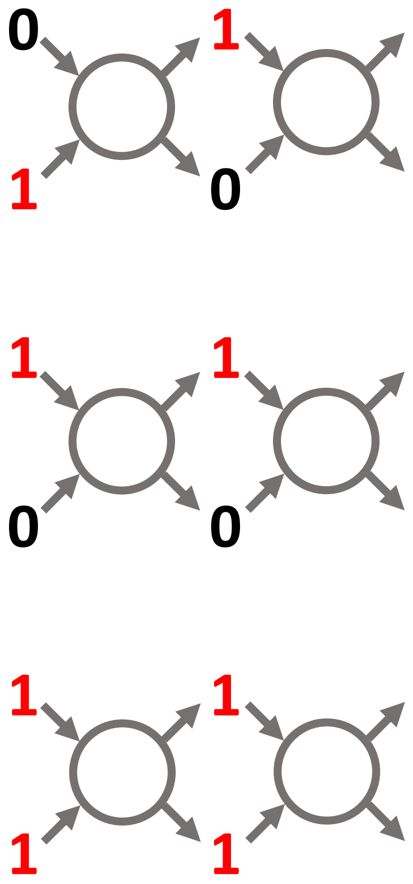
1

1

0

0

Harmonious arrangement



1

1

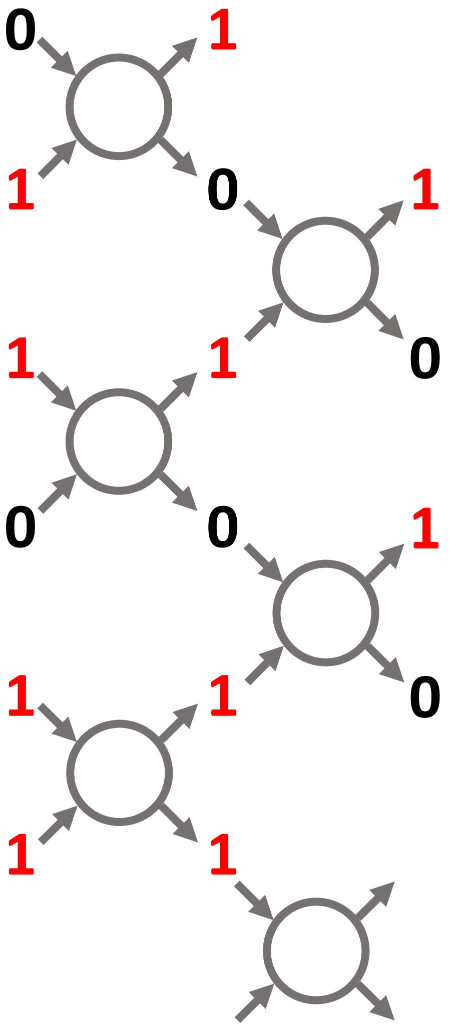
1

1

0

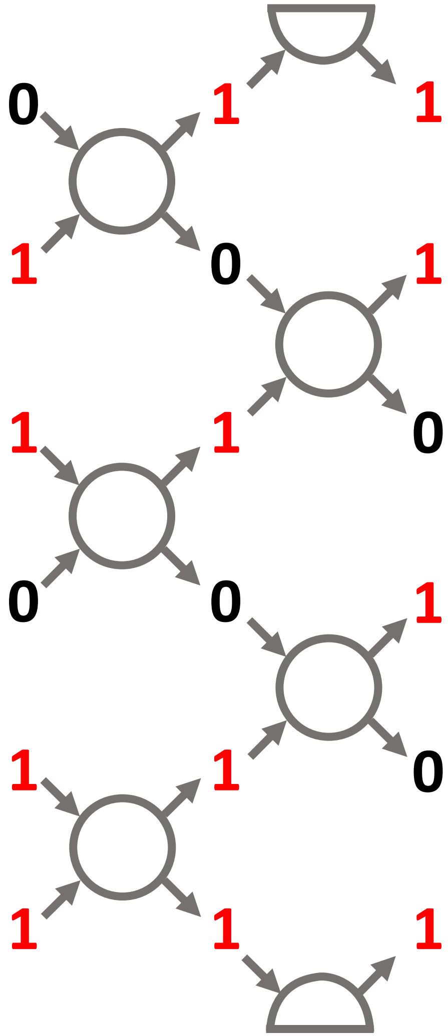
0

Harmonious arrangement



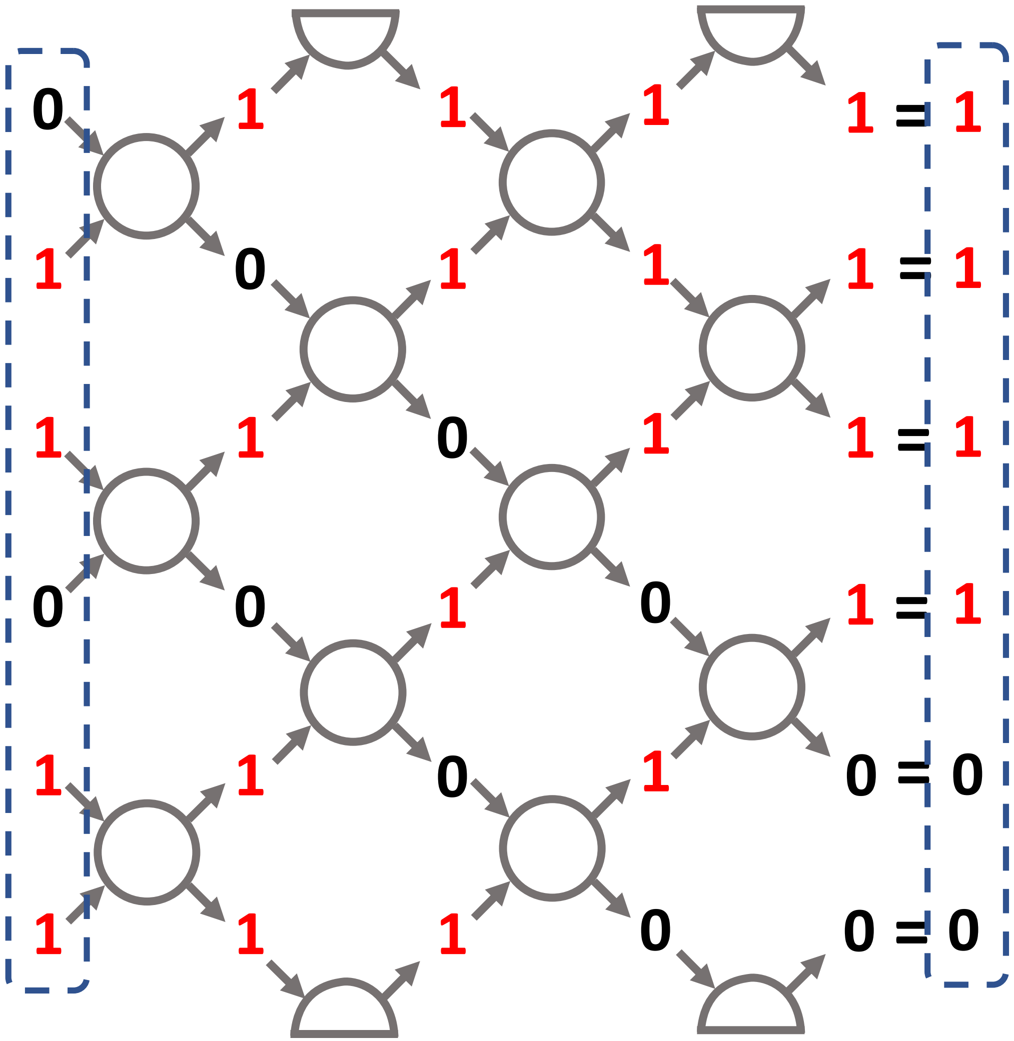
1
1
1
1
0
0

Harmonious arrangement



1
1
1
1
0
0

Harmonious arrangement



a.k.a. sorting

Odd bits

1

0

1

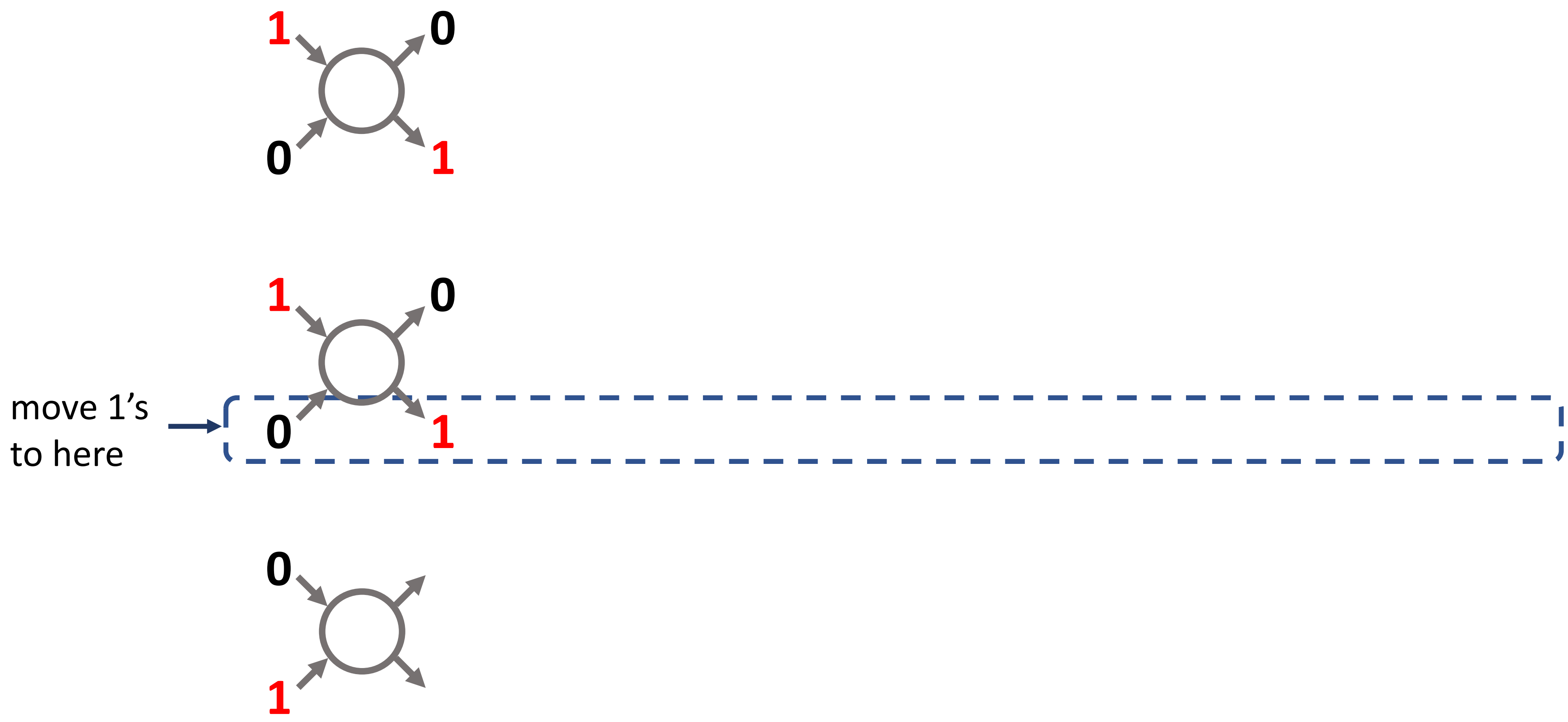
0

0

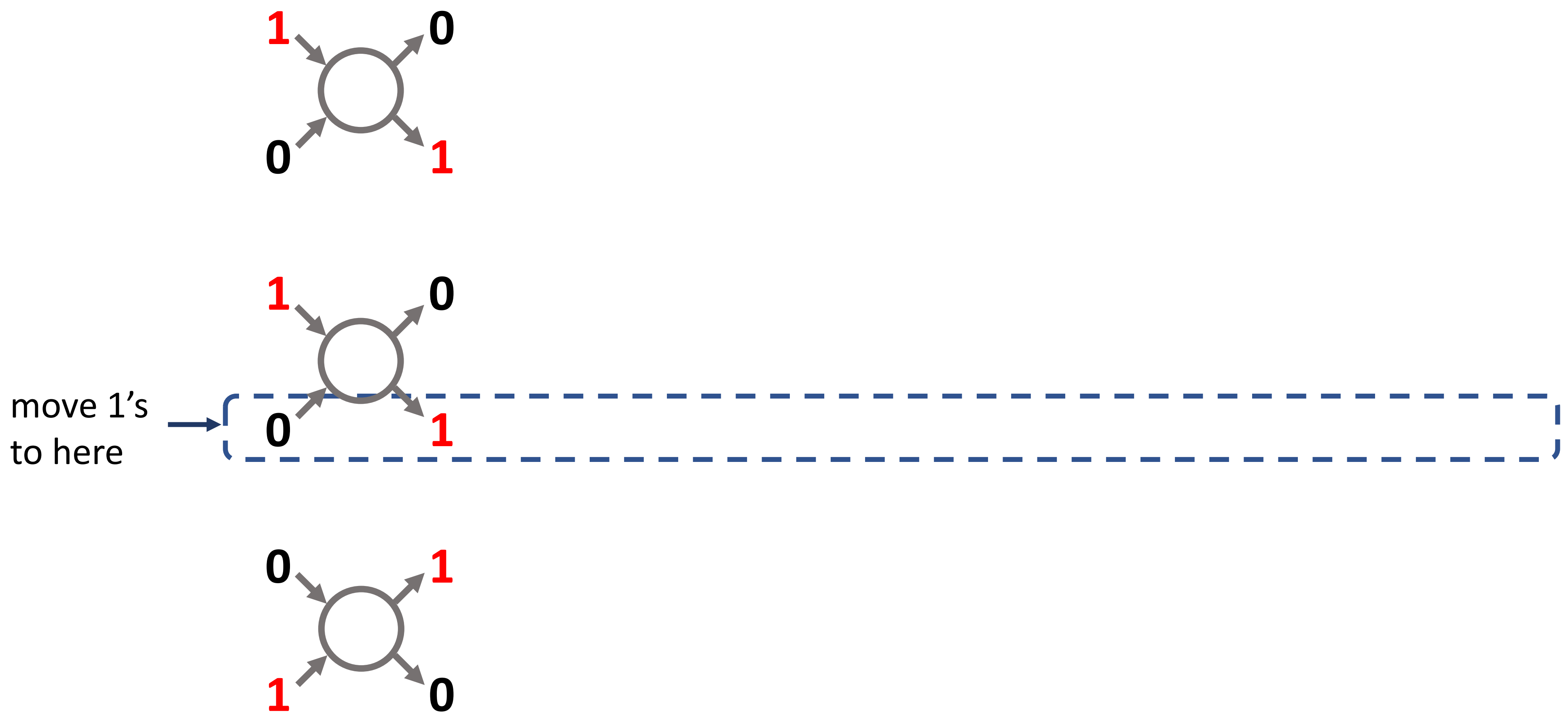
1

Odd bits

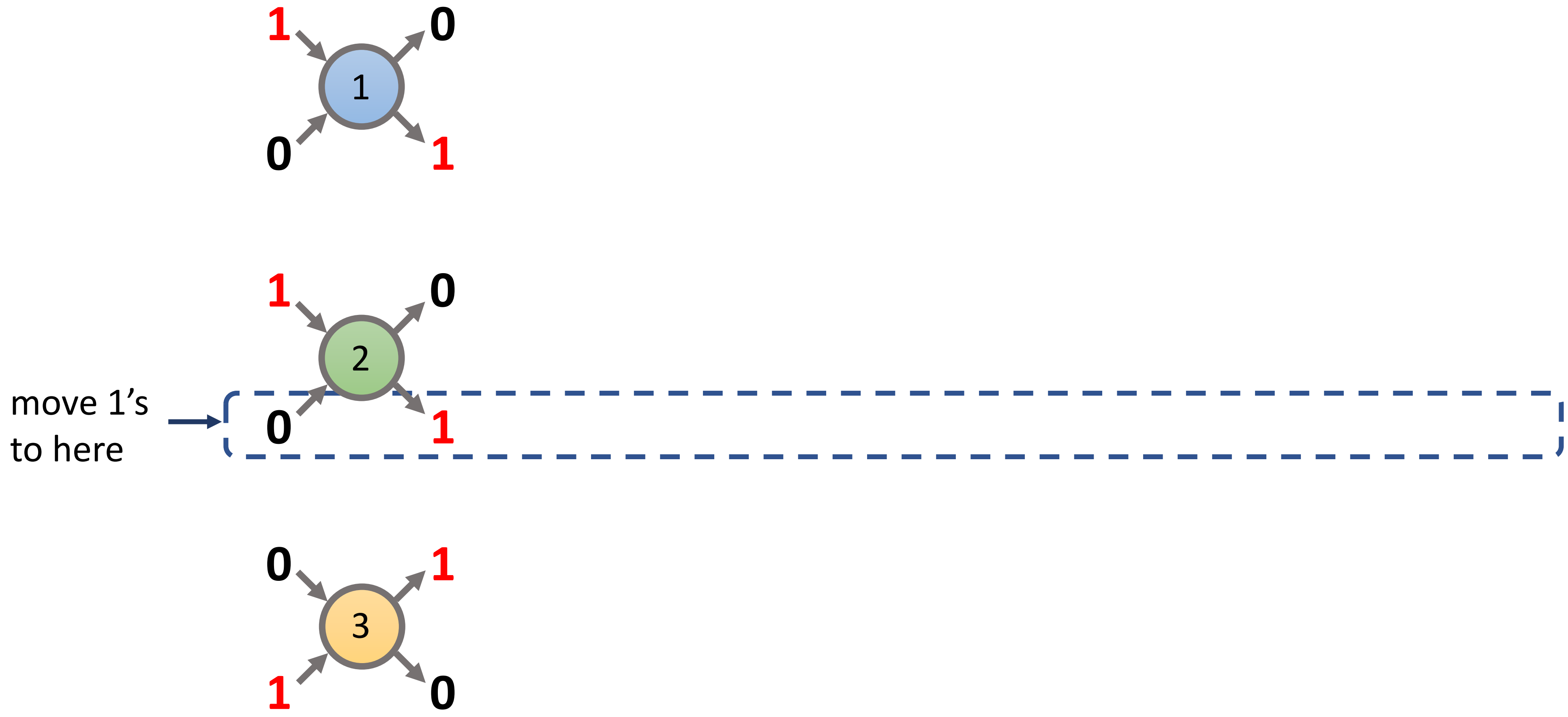
Odd bits



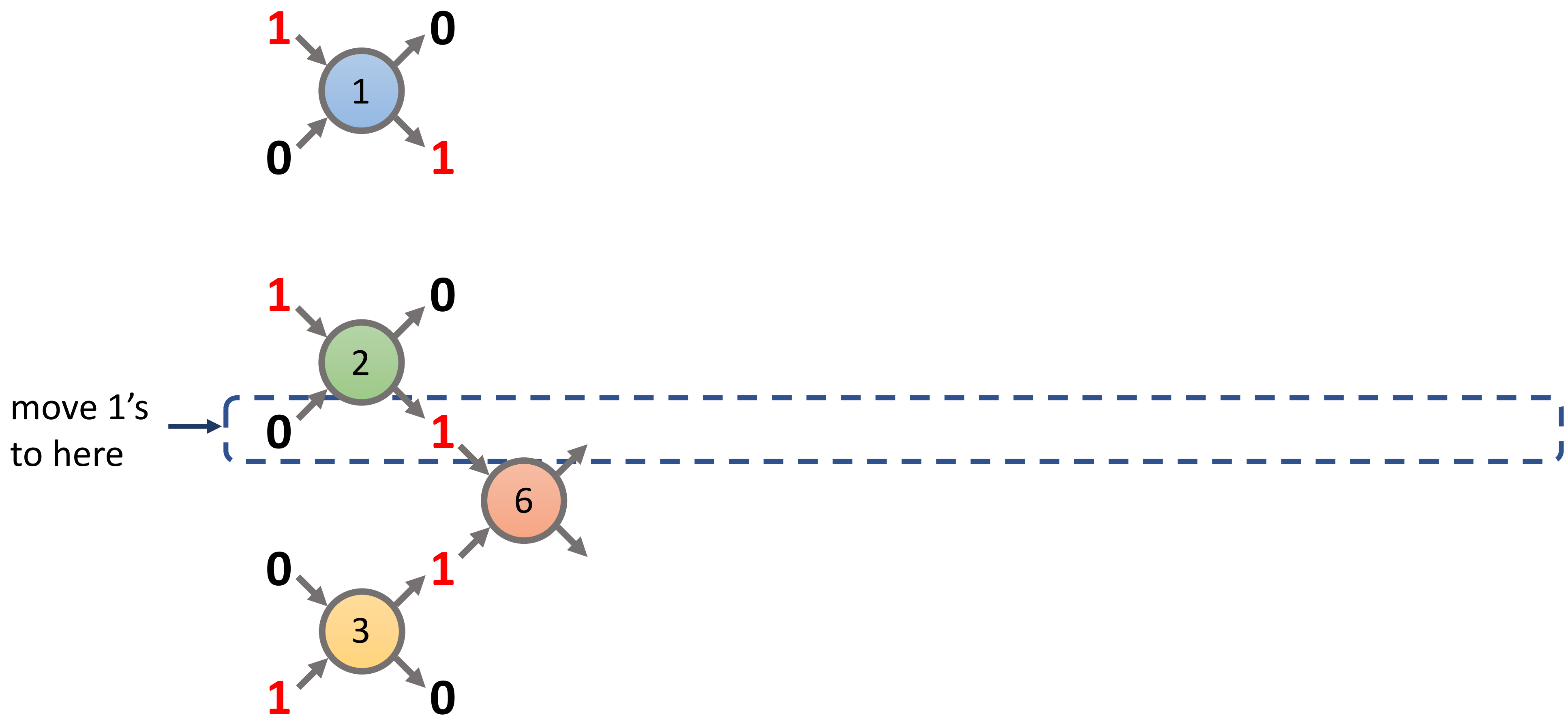
Odd bits



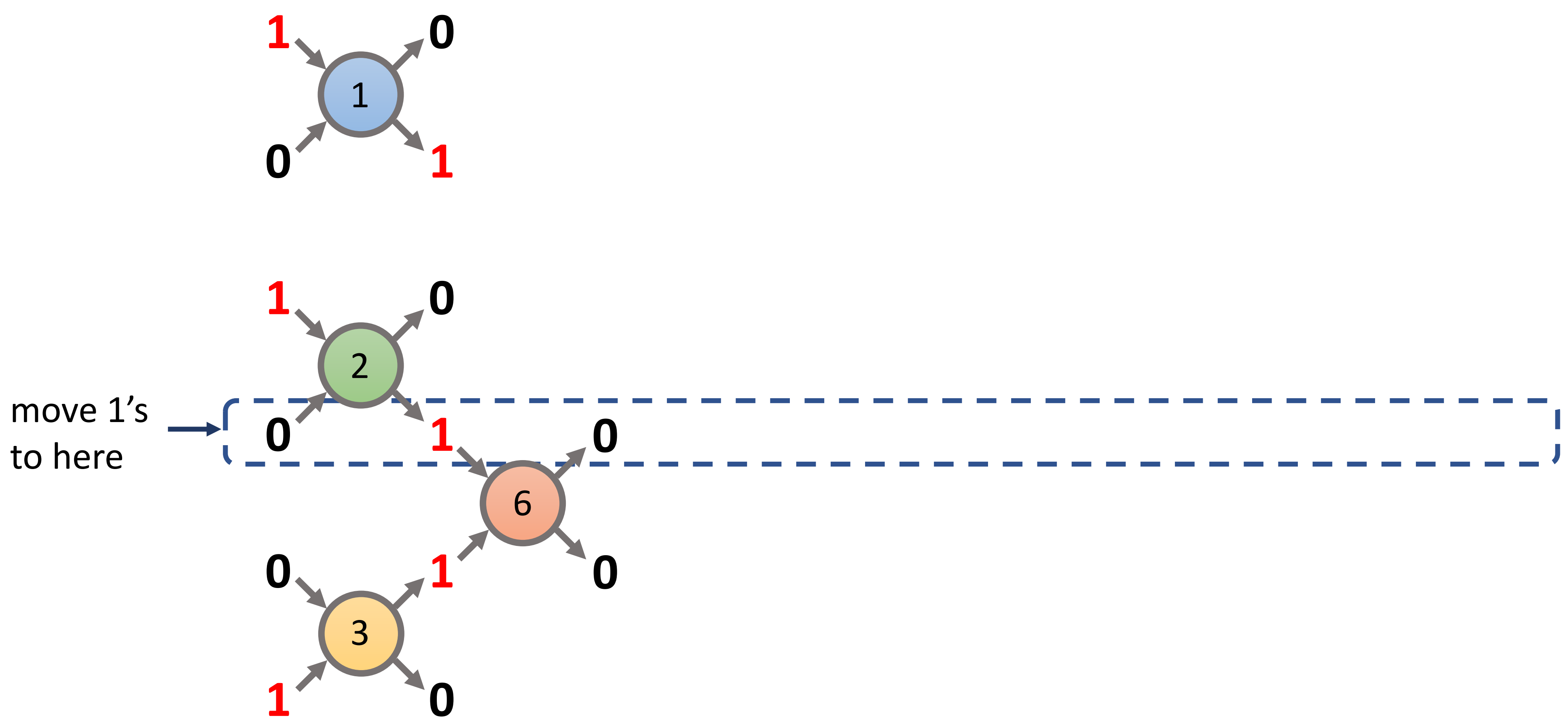
Odd bits



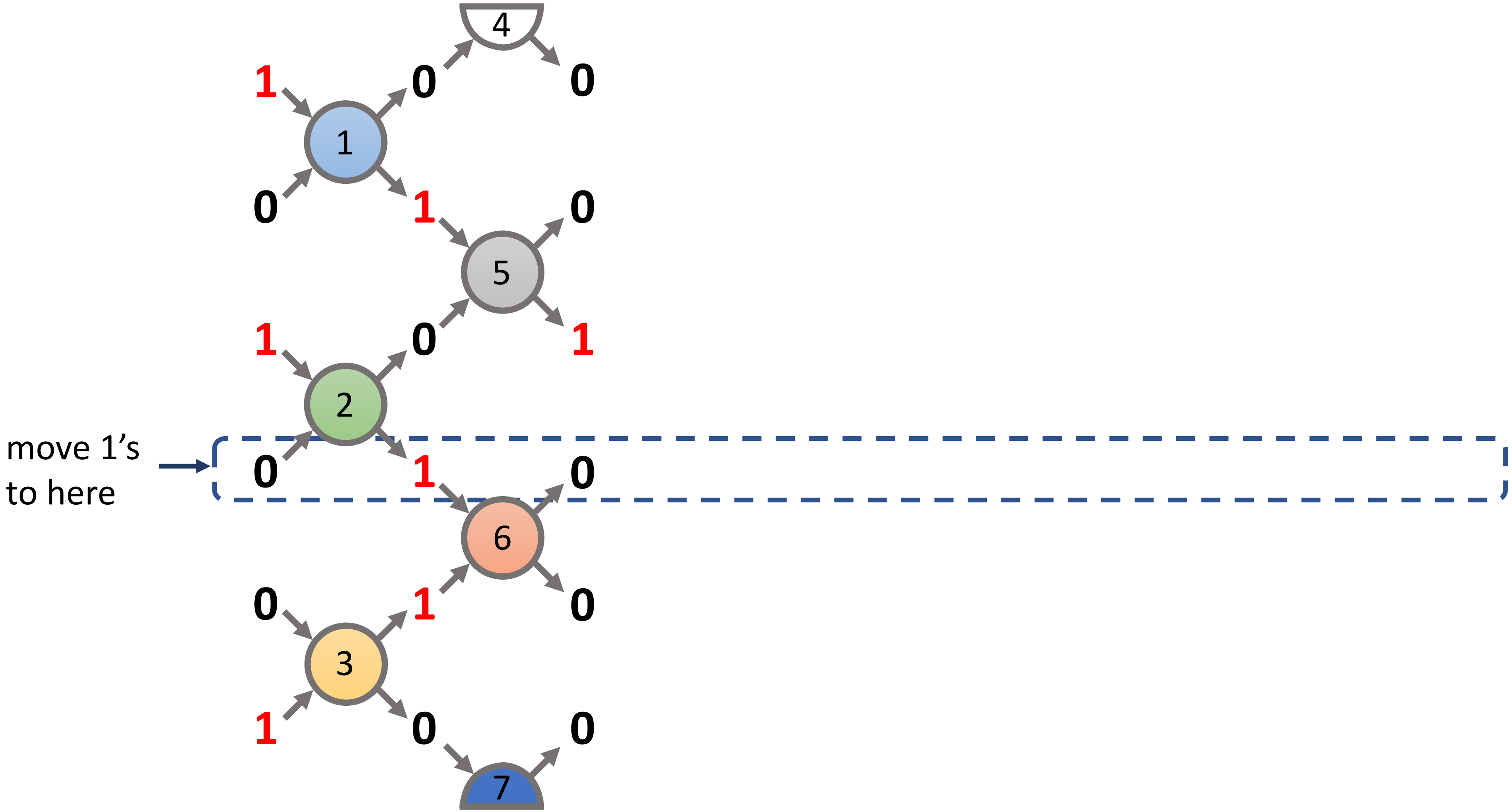
Odd bits



Odd bits

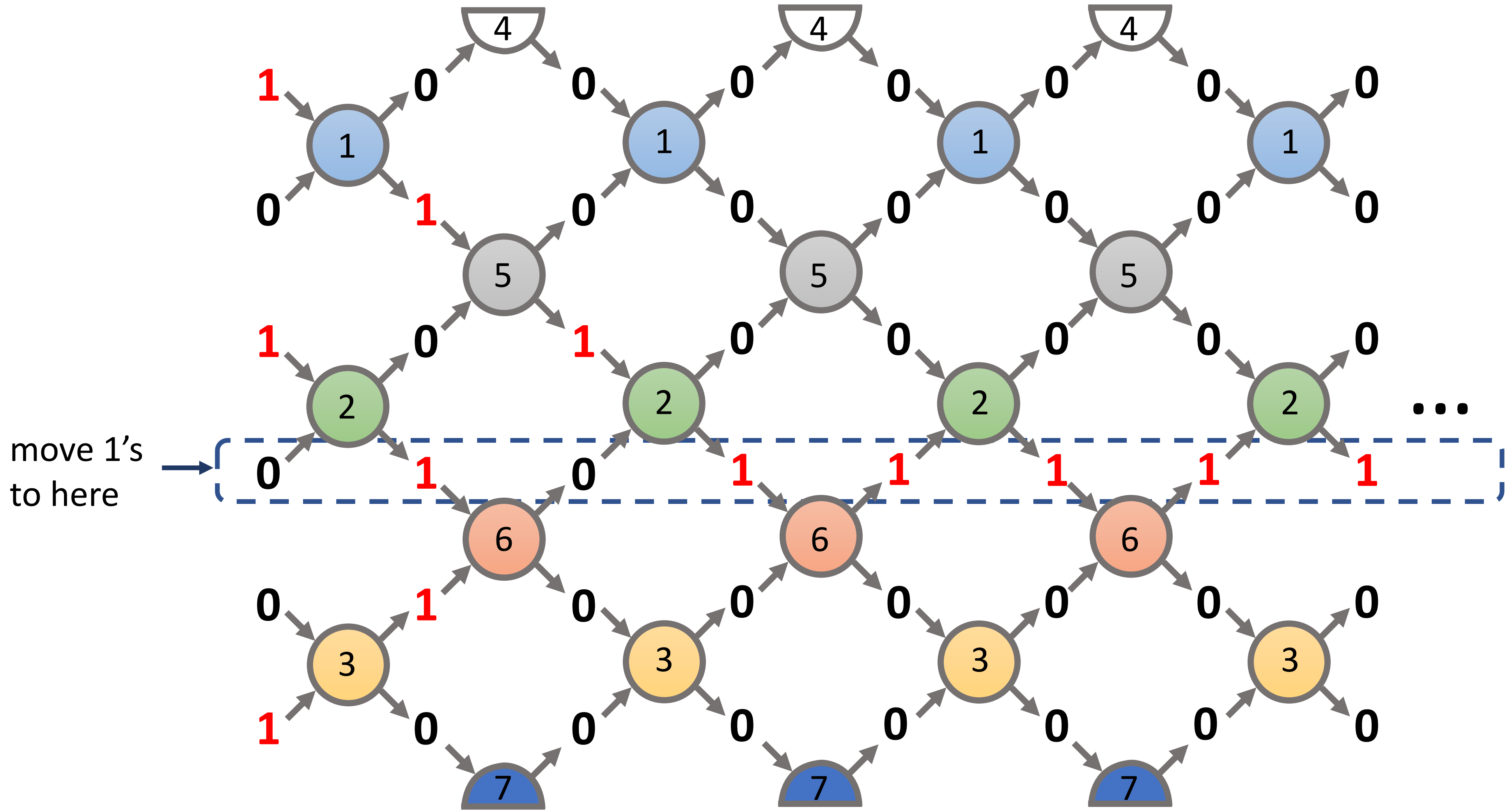


Odd bits



Odd bits

a.k.a. parity



Parity

1

0

1

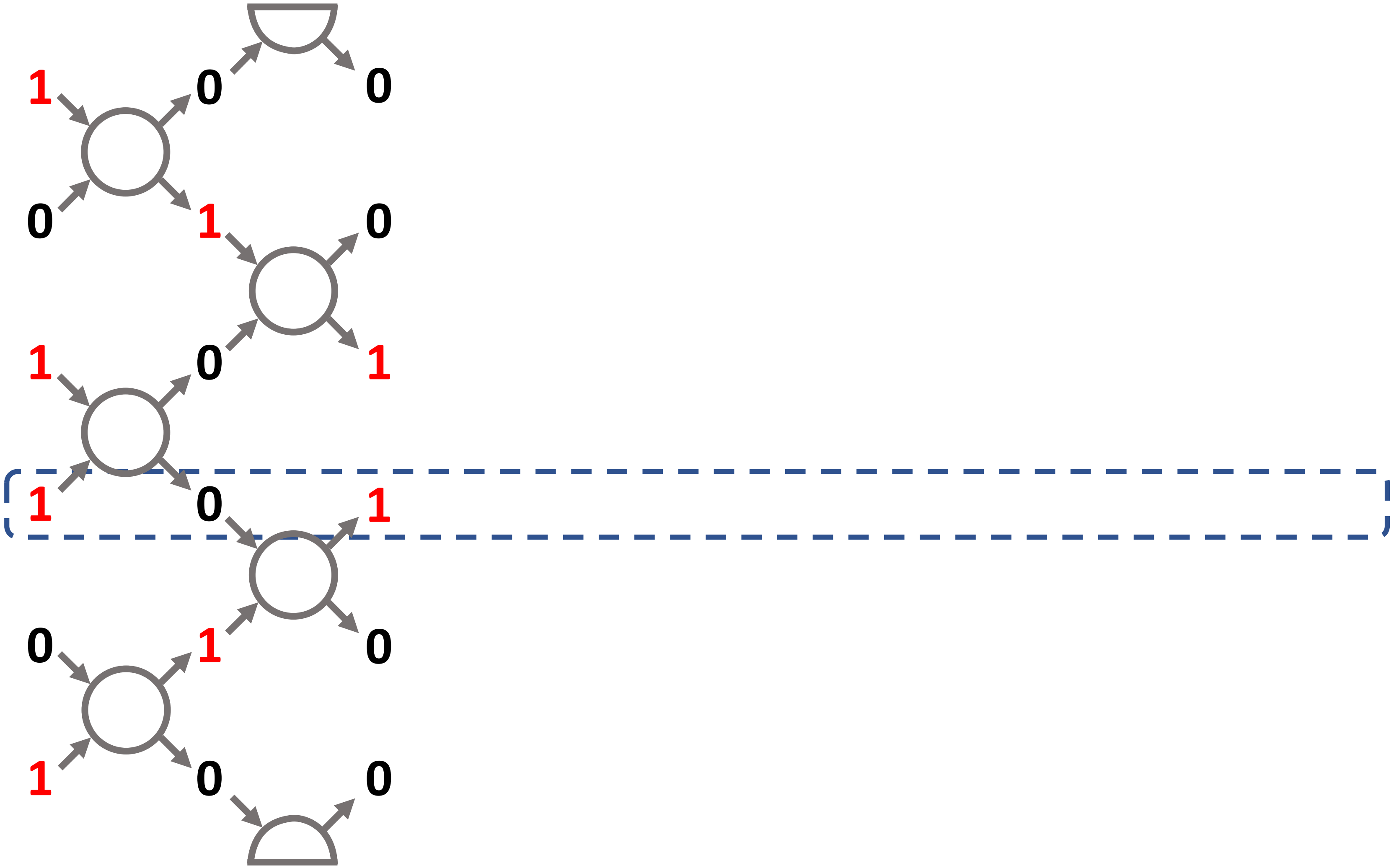
1

0

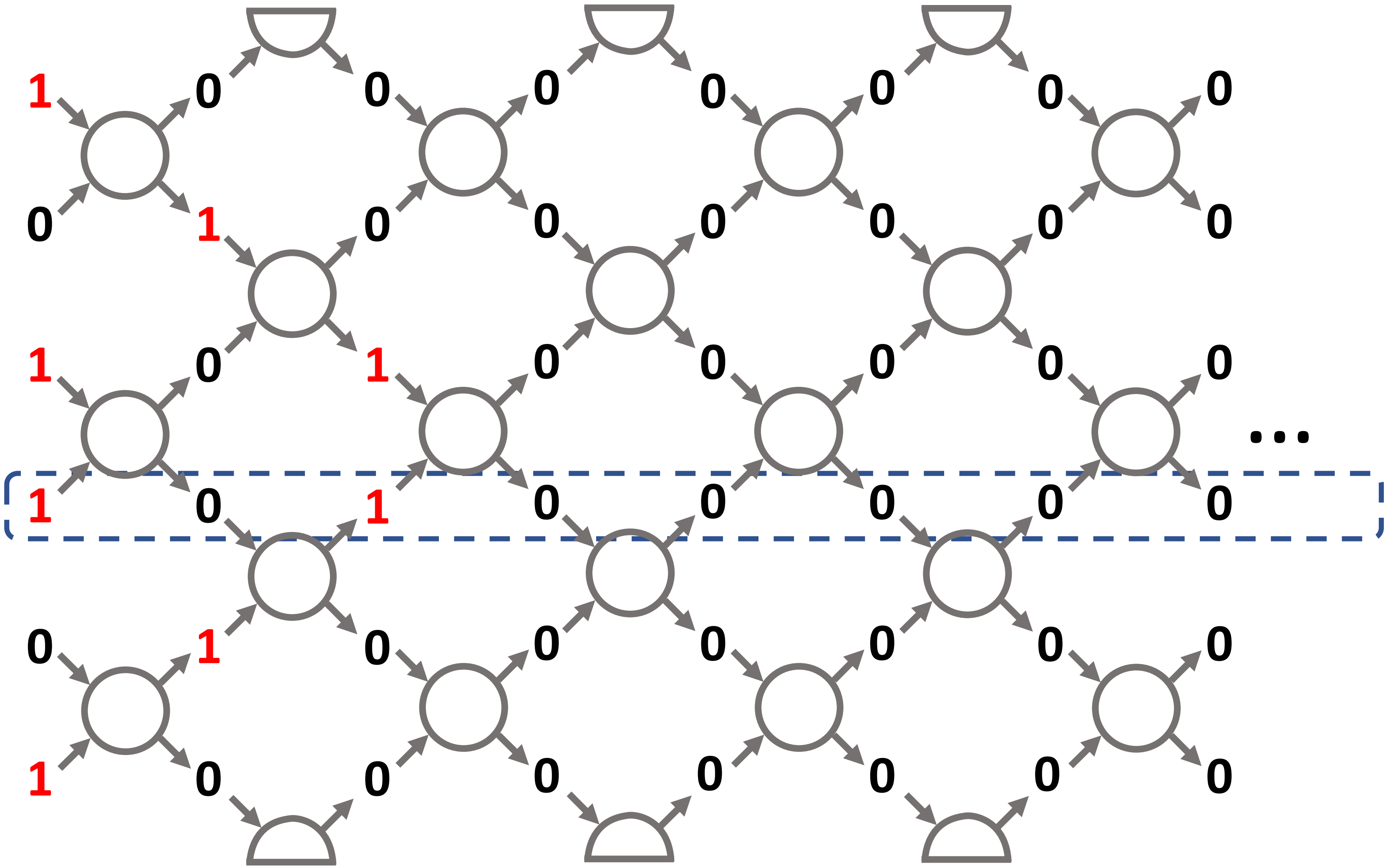
1

Parity

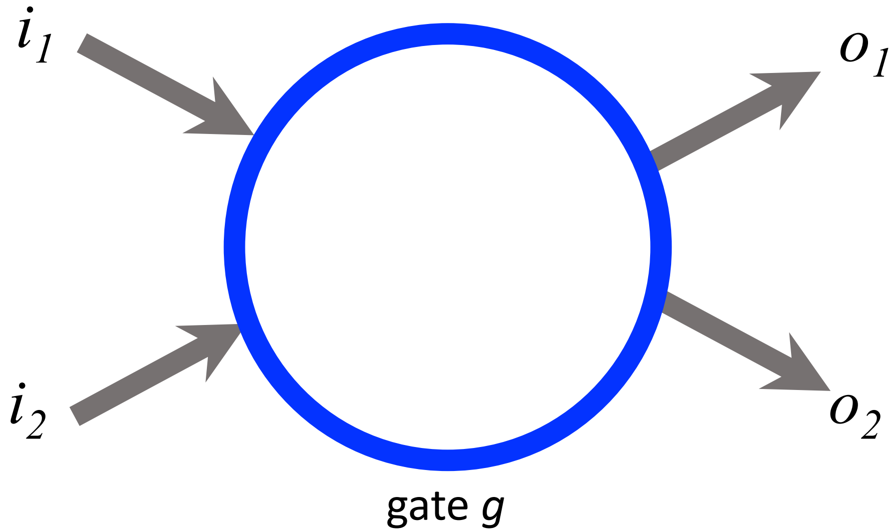
Parity



Parity

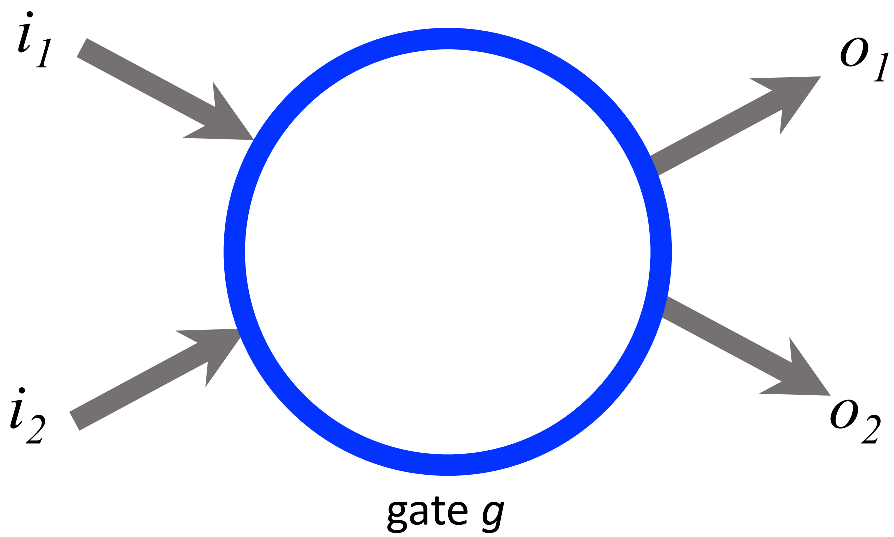


Circuit model



gate g**:** function with two input bits i_1,i_2
and <u>two</u> output bits o_1,o_2

Circuit model

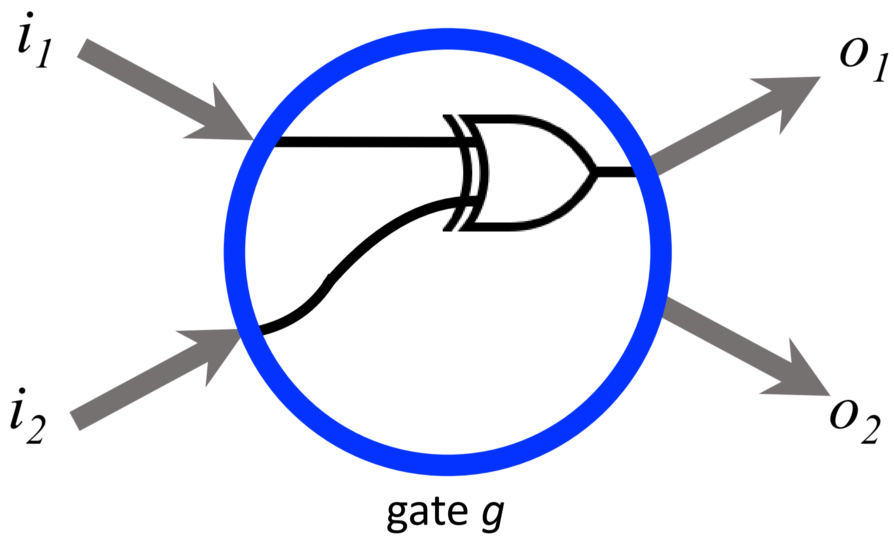


g truth table

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

gate ***g*****:** function with two input bits i_1,i_2
and two output bits o_1,o_2

Circuit model

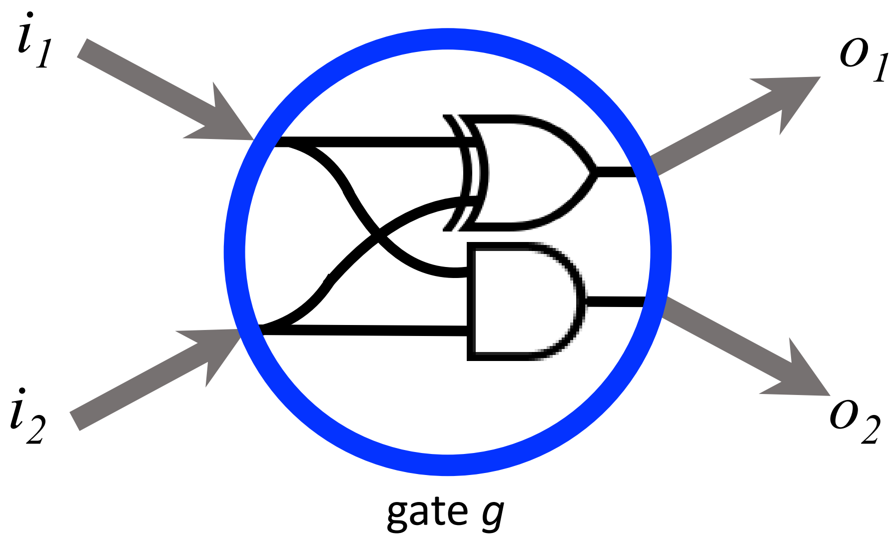


g truth table

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

gate ***g*****:** function with two input bits i_1,i_2
and <u>two</u> output bits o_1,o_2

Circuit model

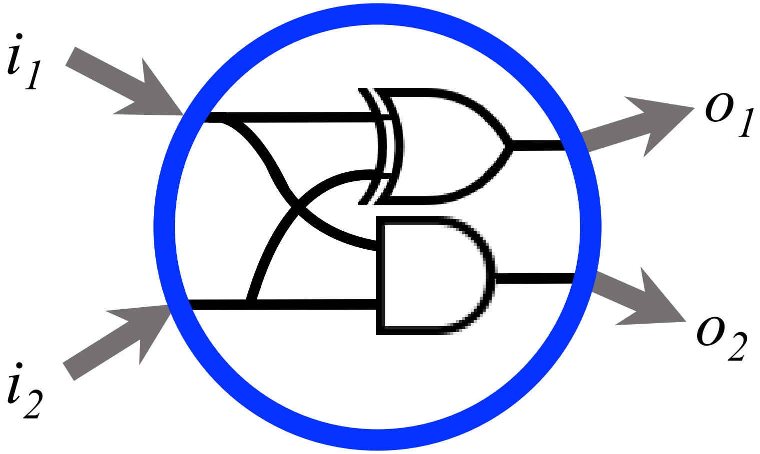


g truth table

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

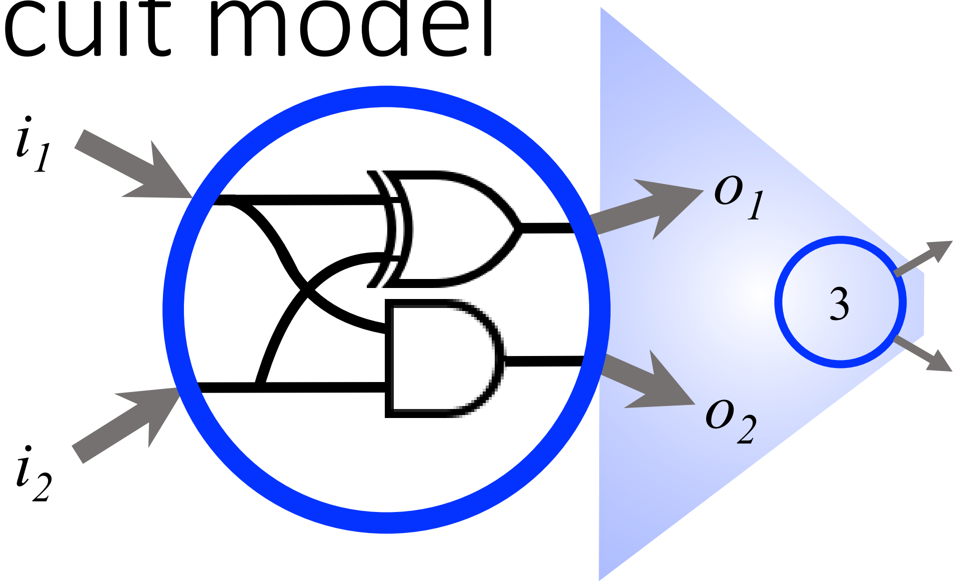
gate ***g*****:** function with two input bits i_1,i_2
and <u>two</u> output bits o_1,o_2

Circuit model



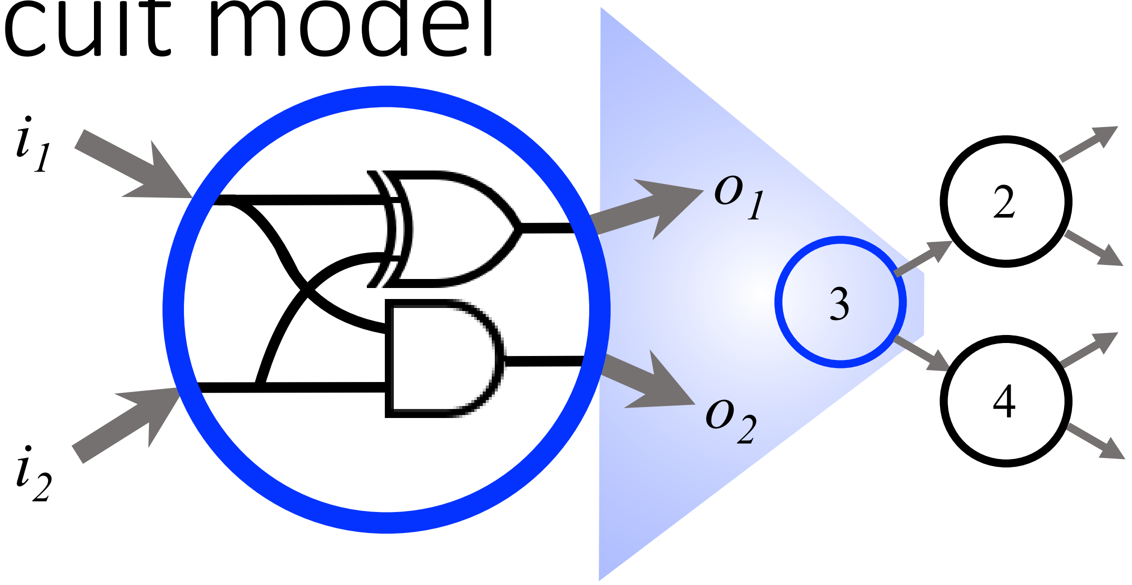
gate $g_{3,XA}(i_1,i_2) = o_1,o_2 = XOR(i_1,i_2),AND(i_1,i_2)$

Circuit model



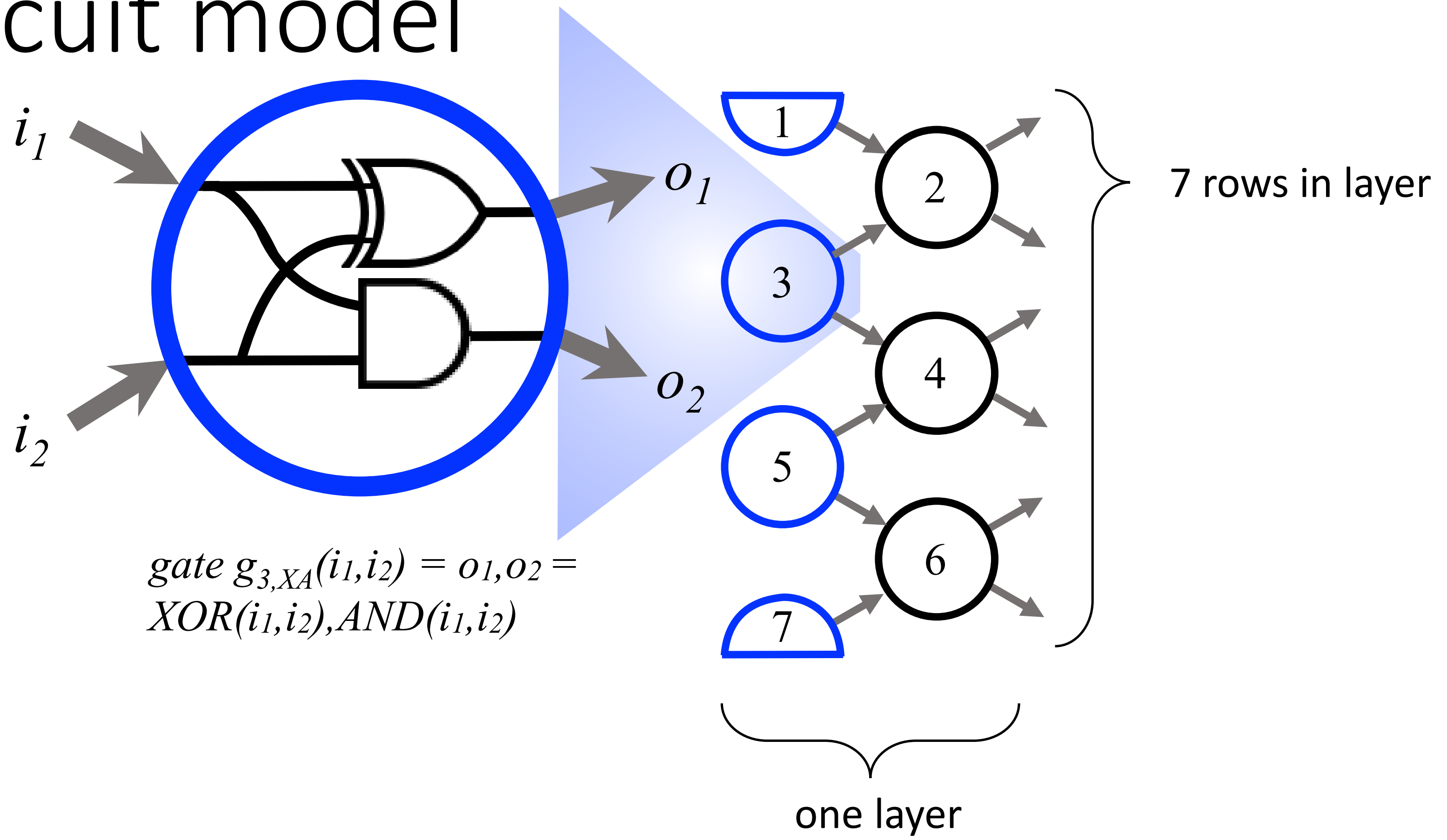
gate $g_{3,XA}(i_1,i_2) = o_1,o_2 = XOR(i_1,i_2),AND(i_1,i_2)$

Circuit model



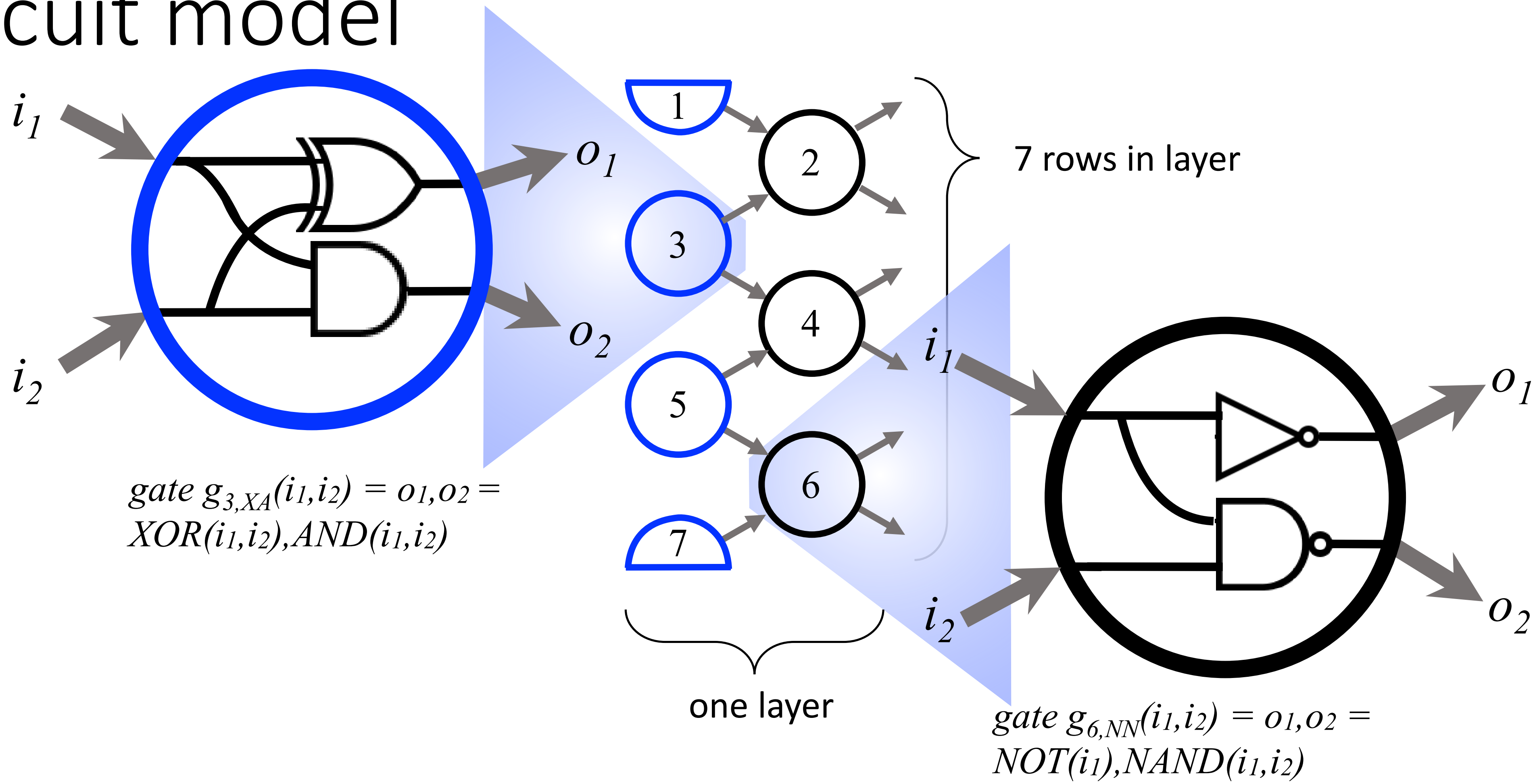
gate $g_{3,XA}(i_1,i_2) = o_1,o_2 = XOR(i_1,i_2),AND(i_1,i_2)$

Circuit model



Gates organized into <u>vertical layers</u> consisting of many <u>rows</u>

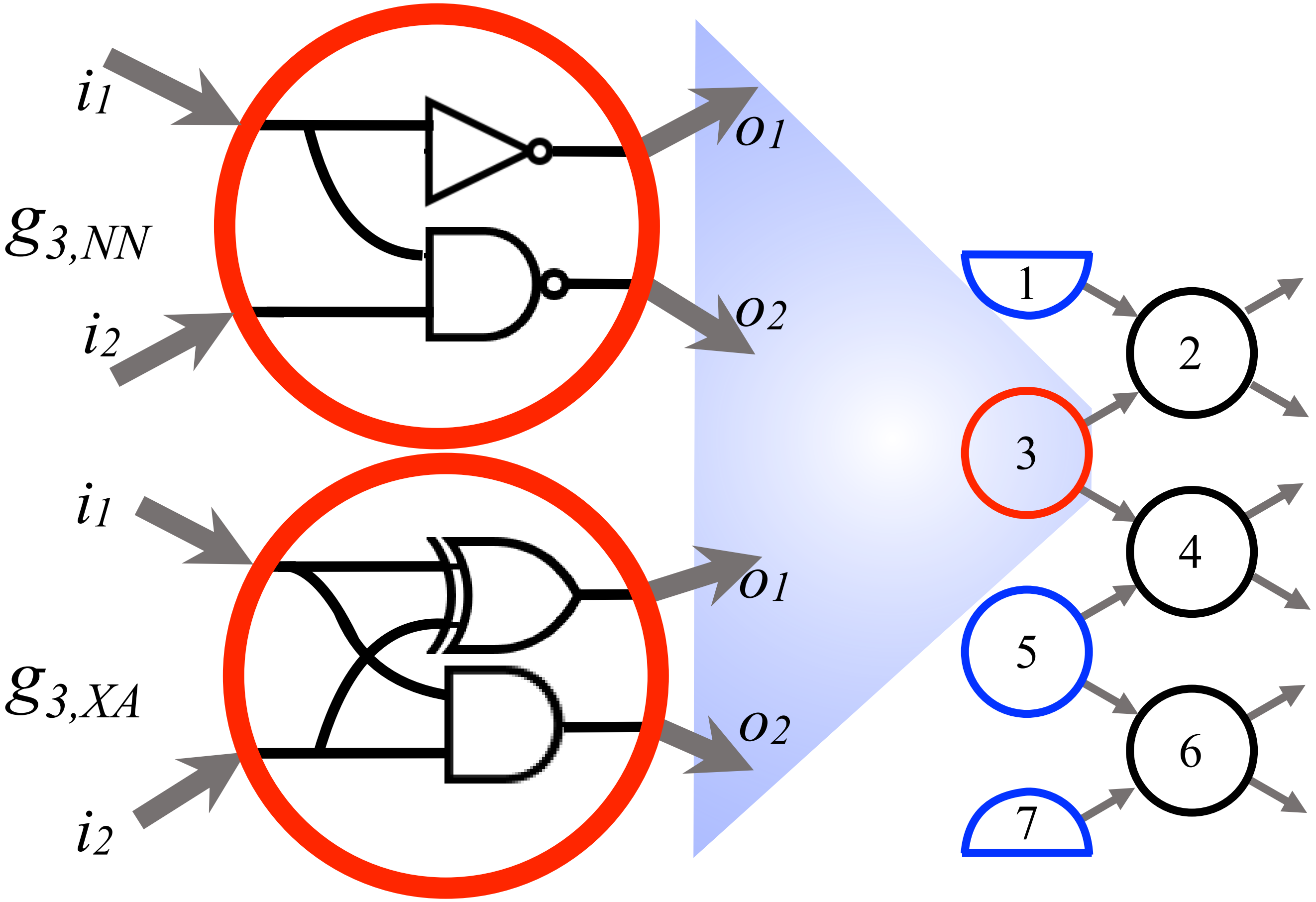
Circuit model



Gates organized into vertical layers consisting of many rows

Possibly different gate types on different rows within a layer

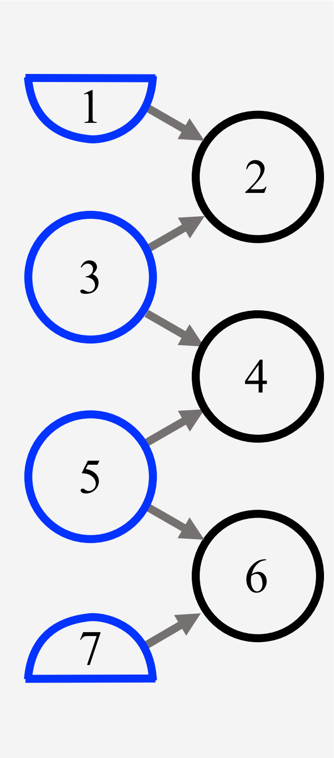
Circuit model



Randomization: Each row may be assigned ≥ 2 gates, with associated probabilities, e.g., $\Pr[g_{3,NN}] = \Pr[g_{3,XA}] = ½$

Circuit model

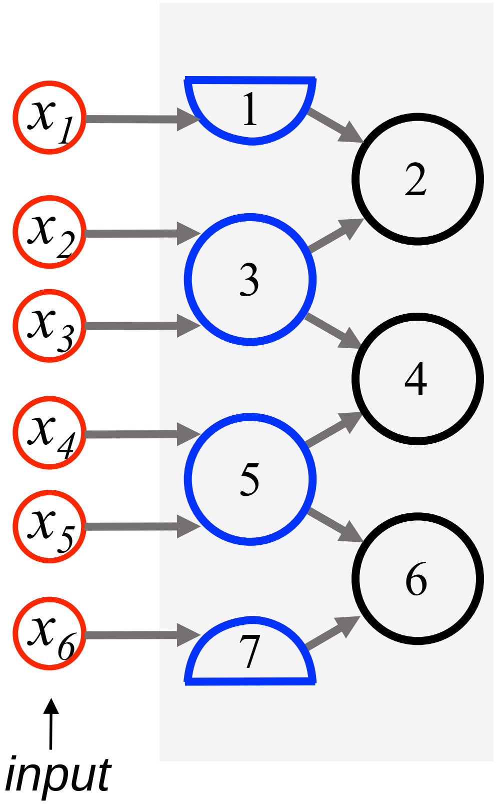
Programmer specifies layer: gates to go in each row



Circuit model

Programmer specifies layer: gates to go in each row

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

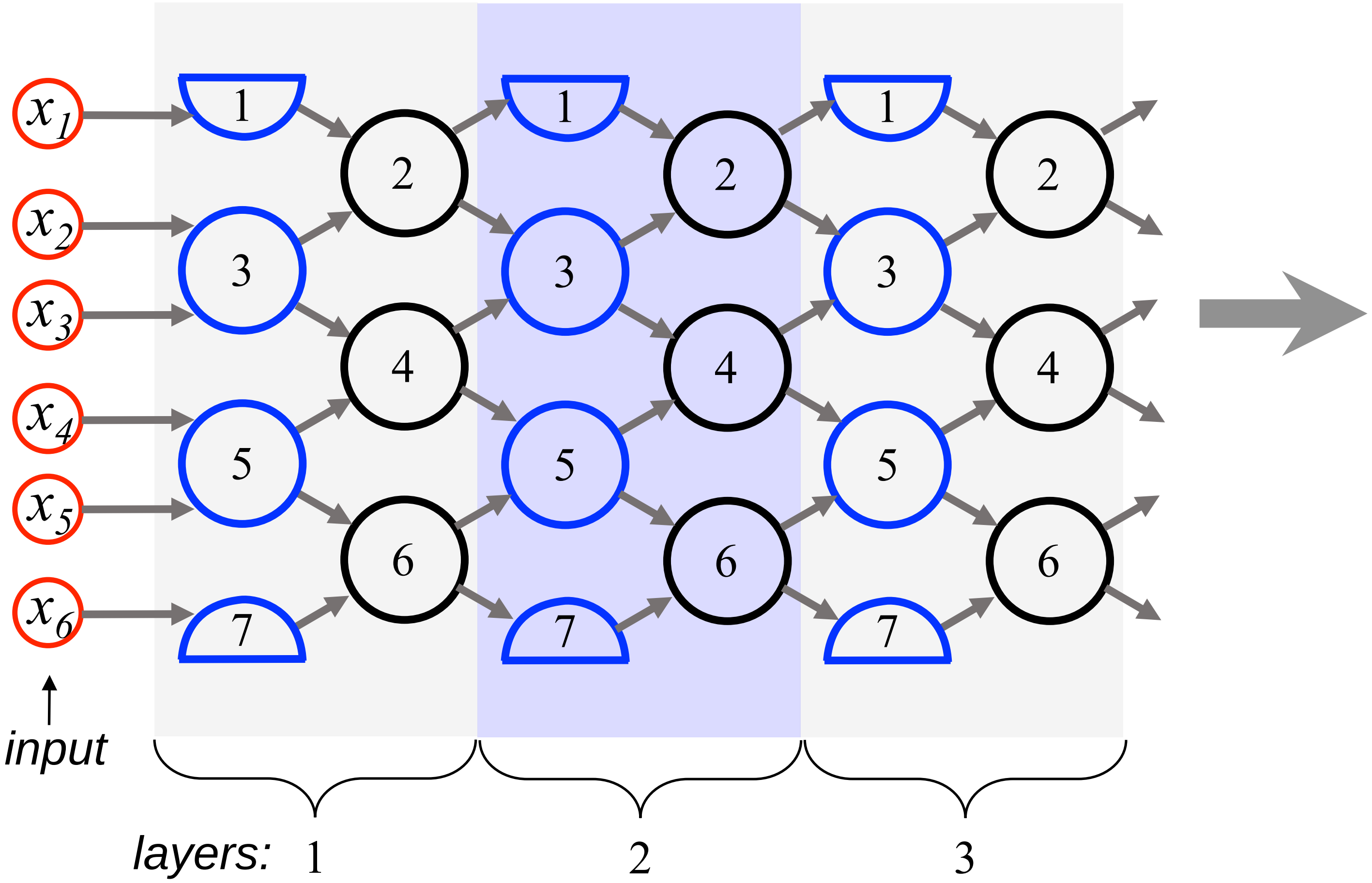


Circuit model

Programmer specifies layer: gates to go in each row

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

Computation flows from inputs to layers 1→2→3→…



Scooped?

US006331788B1

(12) **United States Patent**
Lyke

(10) **Patent No.: US 6,331,788 B1**
(45) **Date of Patent: Dec. 18, 2001**

(54) **SIMPLIFIED CELLULAR ARRAY STRUCTURE FOR PROGRAMMABLE BOOLEAN NETWORKS**

(75) Inventor: **James C. Lyke**, Albuquerque, NM (US)

(73) Assignee: **The United States of America as represented by the Secretary of the Air Force**, Washington, DC (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/681,980**

(22) Filed: **Jul. 3, 2001**

(51) **Int. Cl.**[7] **H03K 19/177**
(52) **U.S. Cl.** **326/39**; 326/41
(58) **Field of Search** 326/37–41

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,069,490 * 5/2000 Ochotta et al. 326/41
6,114,873 * 9/2000 Sahraoui et al. 326/39
6,122,720 * 9/2000 Cliff 712/15
6,215,327 * 4/2001 Lyke 326/41

OTHER PUBLICATIONS

Christopher Moore and Arthur A. Drisko, "Algebraic Properties of the Block Transformation of Cellular Automata", Complex Systems, vol. 10, No. 3,1996, 185–194.

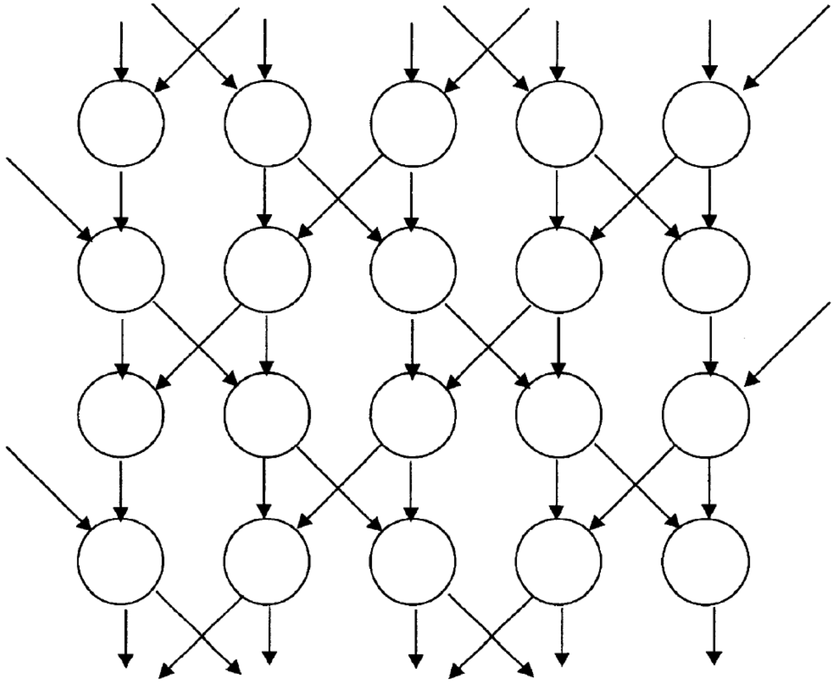
* cited by examiner

Primary Examiner—Michael Tokar
Assistant Examiner—Don Phu Le
(74) *Attorney, Agent, or Firm*—Kenneth E. Callahan

(57) **ABSTRACT**

PRS07010 A simplified implementation of molecular field programmable gate arrays described in U.S. Pat. No. 6,215,327, reducing the complexity of a single site in a tiled array template to that of a 2-input lookup table.

2 Claims, 11 Drawing Sheets



BACKGROUND OF THE INVENTION

The invention relates to field programmable gate array circuits used in digital circuit design, and in particular, to a simplified architectural design for their implementation on a molecular level related to U.S. Pat. No. 6,215,327.

U.S. Patent Dec. 18, 2001 Sheet 9 of 11 US 6,331,788 B1

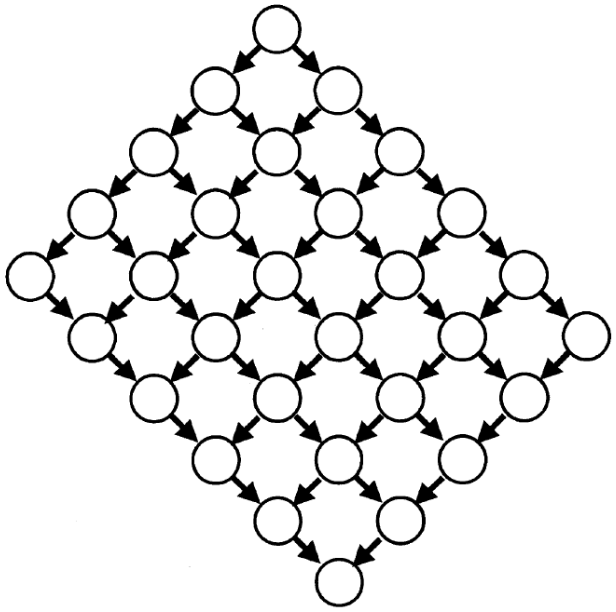


FIG. 9

U.S. Patent Dec. 18, 2001 Sheet 10 of 11 US 6,331,788 B1

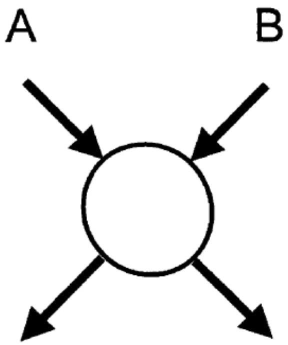


FIG. 10a

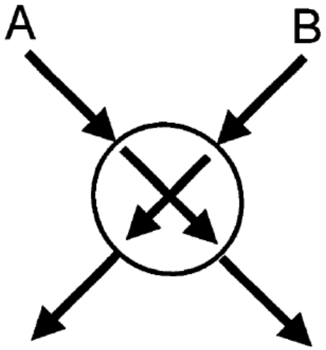


FIG. 10b

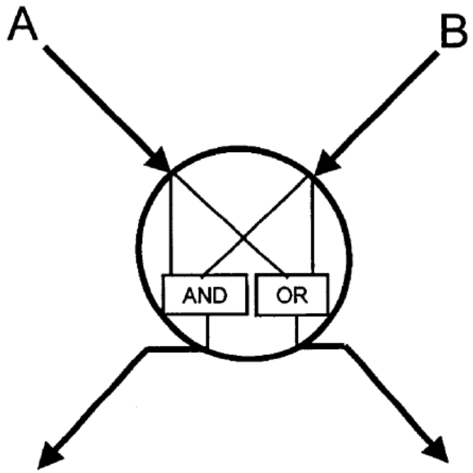


FIG. 10c

Scooped?

US006331788B1

(12) **United States Patent**
Lyke

(10) **Patent No.: US 6,331,788 B1**
(45) **Date of Patent: Dec. 18, 2001**

(54) **SIMPLIFIED CELLULAR ARRAY STRUCTURE FOR PROGRAMMABLE BOOLEAN NETWORKS**

(75) Inventor: **James C. Lyke**, Albuquerque, NM (US)

(73) Assignee: **The United States of America as represented by the Secretary of the Air Force**, Washington, DC (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/681,980**

(22) Filed: **Jul. 3, 2001**

(51) **Int. Cl.**[7] **H03K 19/177**
(52) **U.S. Cl.** **326/39**; 326/41
(58) **Field of Search** 326/37–41

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,069,490 * 5/2000 Ochotta et al. 326/41
6,114,873 * 9/2000 Sahraoui et al. 326/39
6,122,720 * 9/2000 Cliff 712/15
6,215,327 * 4/2001 Lyke 326/41

OTHER PUBLICATIONS

Christopher Moore and Arthur A. Drisko, "Algebraic Properties of the Block Transformation of Cellular Automata", Complex Systems, vol. 10, No. 3,1996, 185–194.

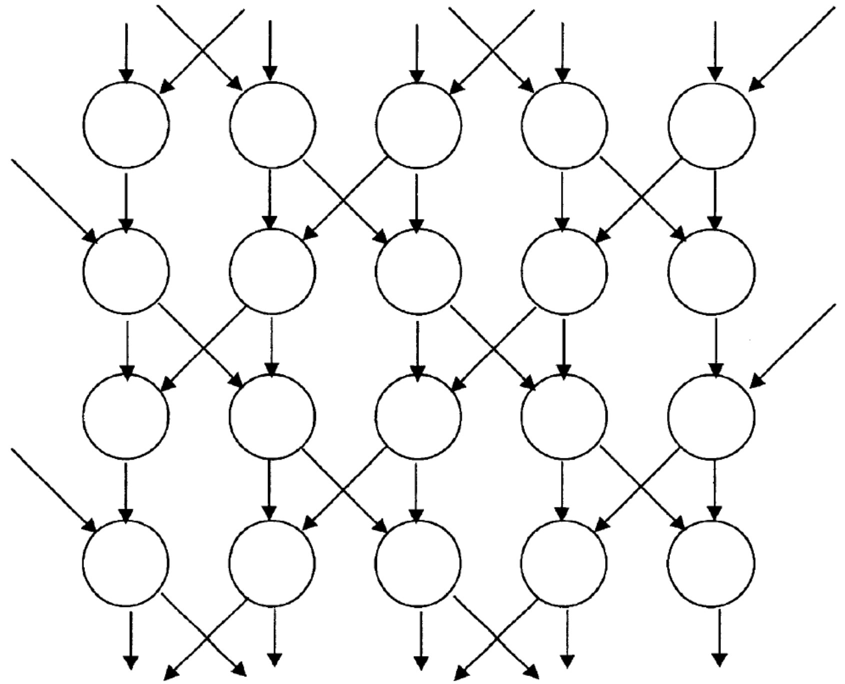
* cited by examiner

Primary Examiner—Michael Tokar
Assistant Examiner—Don Phu Le
(74) *Attorney, Agent, or Firm*—Kenneth E. Callahan

(57) **ABSTRACT**

PRS07010 A simplified implementation of molecular field programmable gate arrays described in U.S. Pat. No. 6,215,327, reducing the complexity of a single site in a tiled array template to that of a 2-input lookup table.

2 Claims, 11 Drawing Sheets



BACKGROUND OF THE INVENTION

The invention relates to field programmable gate array circuits used in digital circuit design, and in particular, to a simplified architectural design for their implementation on a molecular level related to U.S. Pat. No. 6,215,327.

U.S. Patent Dec. 18, 2001 Sheet 9 of 11 US 6,331,788 B1

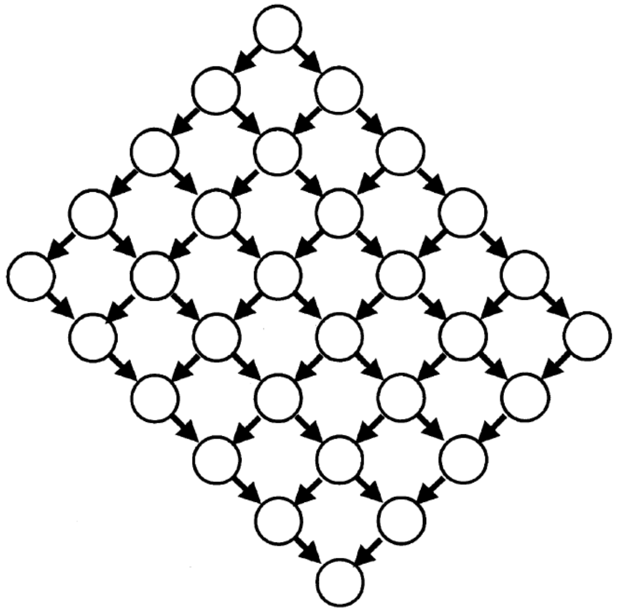


FIG. 9

U.S. Patent Dec. 18, 2001 Sheet 10 of 11 US 6,331,788 B1

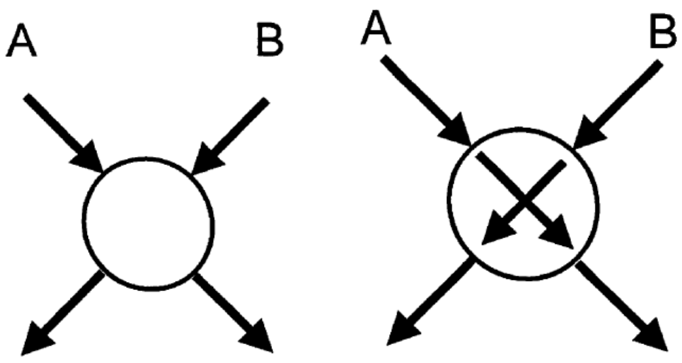


FIG. 10a FIG. 10b

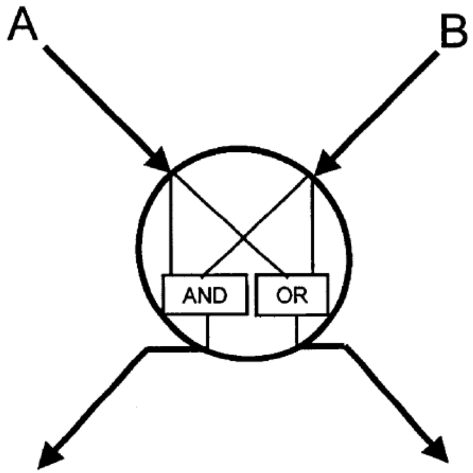
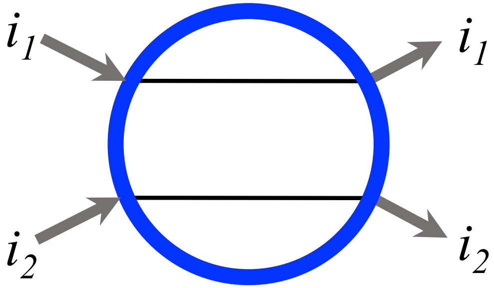


FIG. 10c

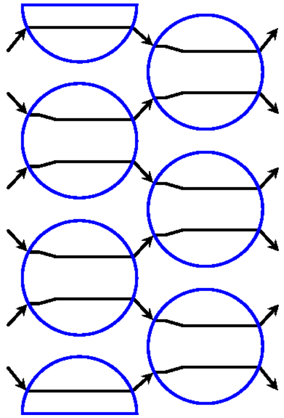
Example circuits with same gate in every row

COPY

COPY gates

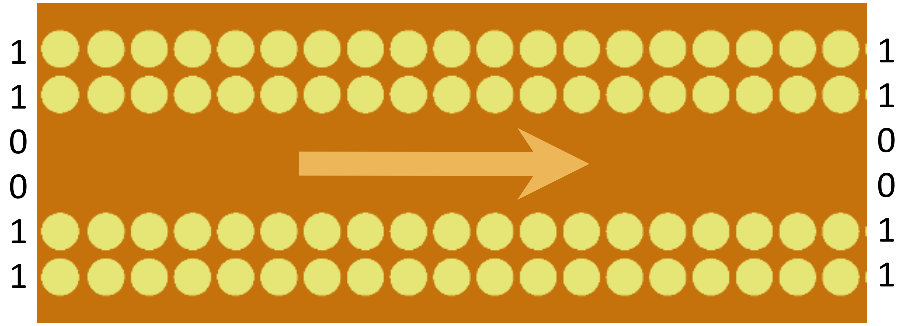


i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

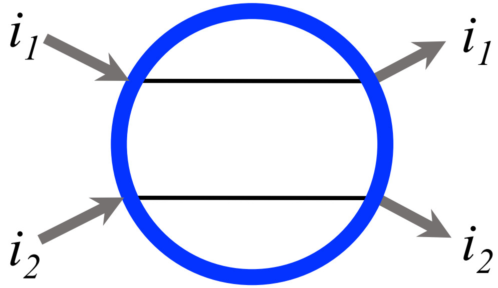


Example circuits with same gate in every row

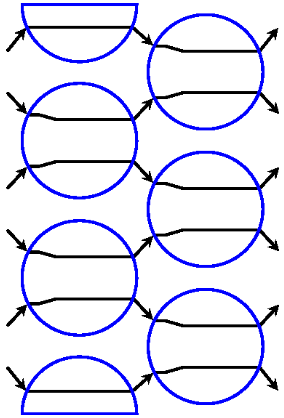
COPY



COPY gates

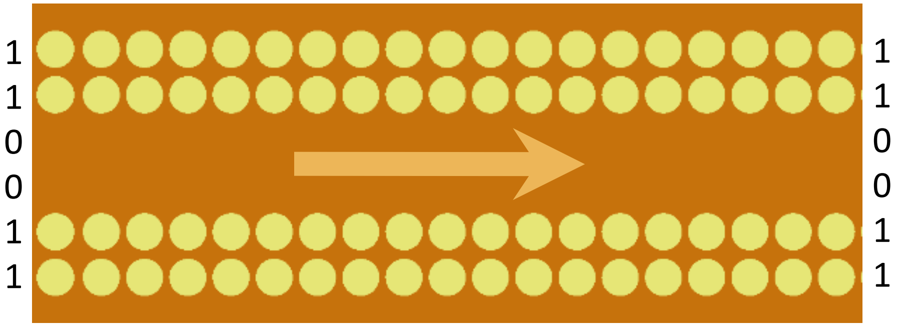


i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

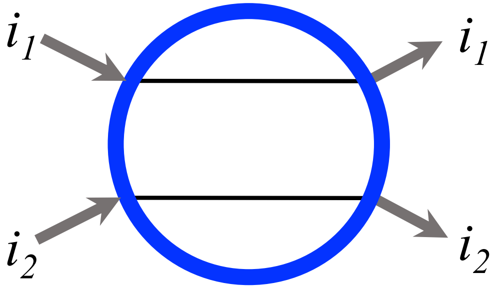


Example circuits with same gate in every row

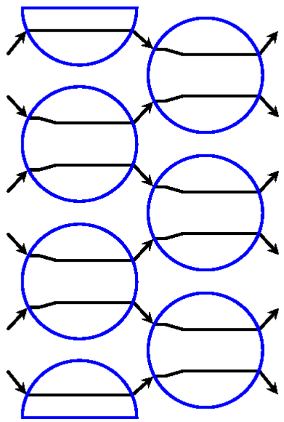
Copy



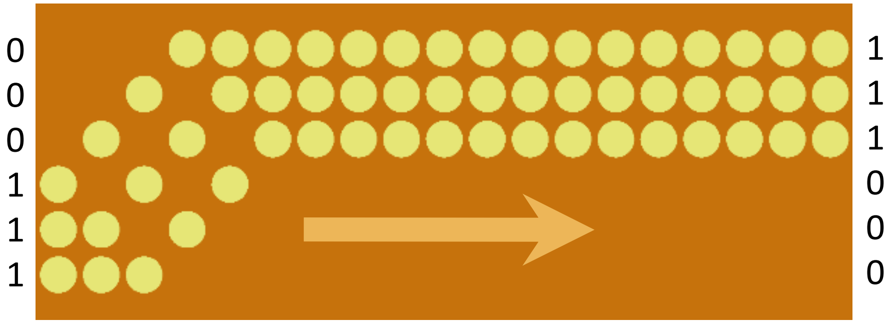
Copy gates



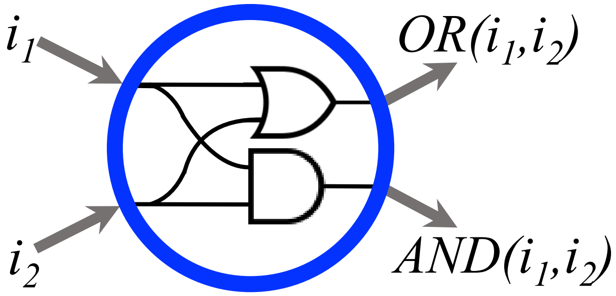
i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1



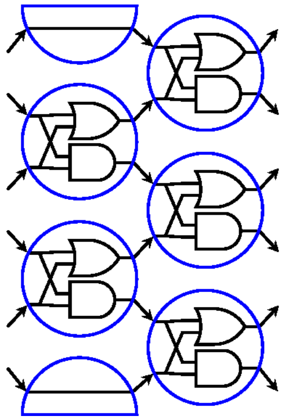
Sorting



Sorting gates

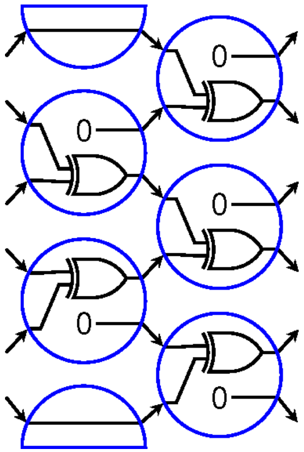


i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



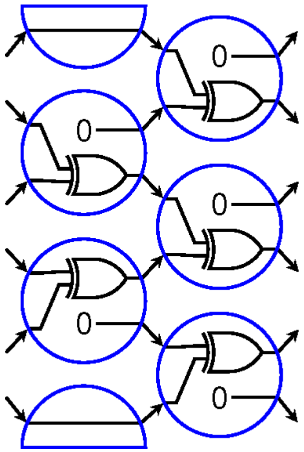
Example circuits with different gates in each row

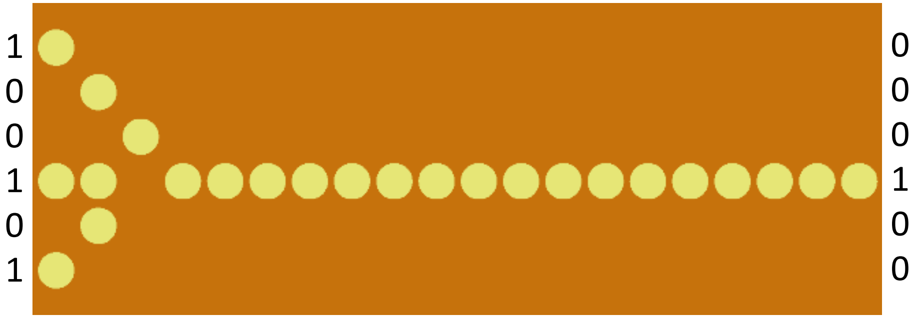
PARITY



Example circuits with different gates in each row

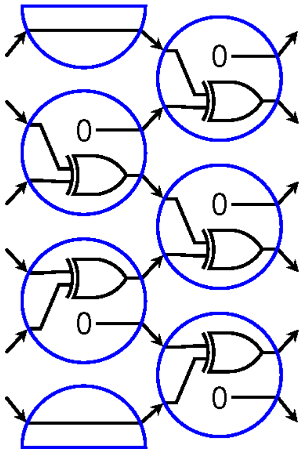
PARITY

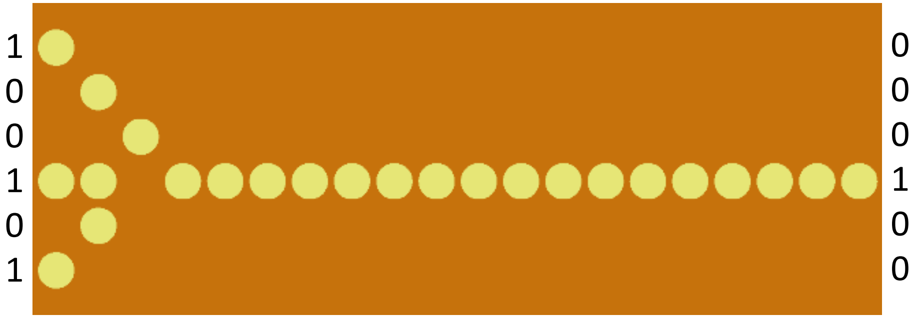




Example circuits with different gates in each row

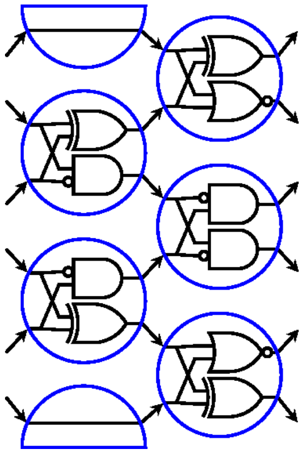
PARITY

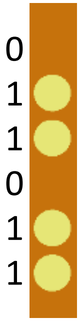




MULTIPLEOF3

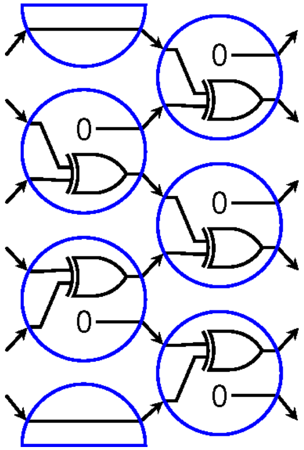
011011_2

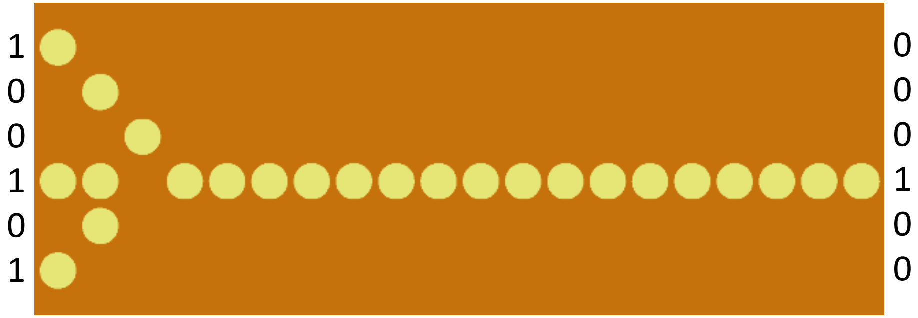




Example circuits with different gates in each row

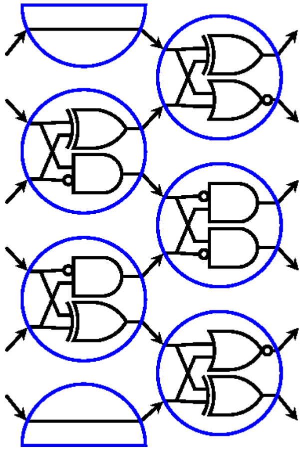
PARITY

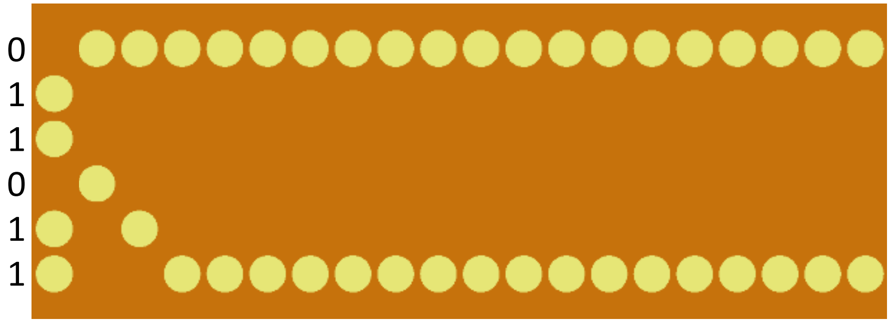




MULTIPLEOF3

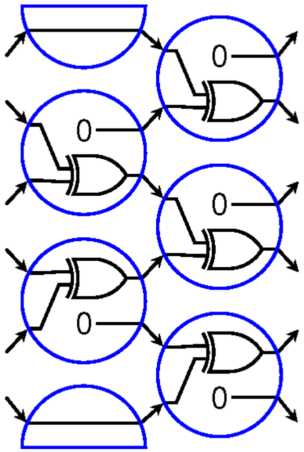
$011011_2 = 27_{10} = 3\cdot 9$

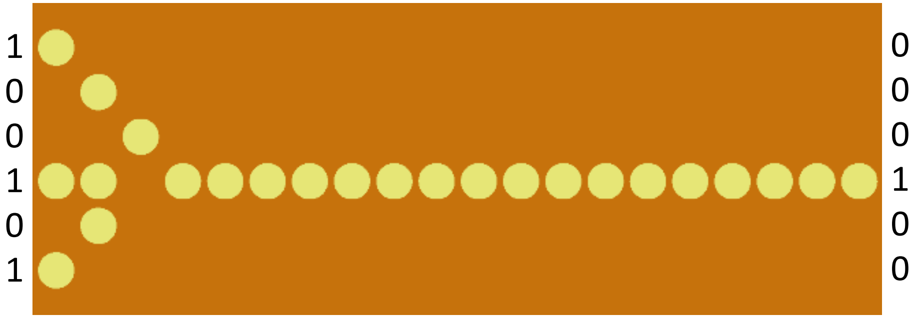




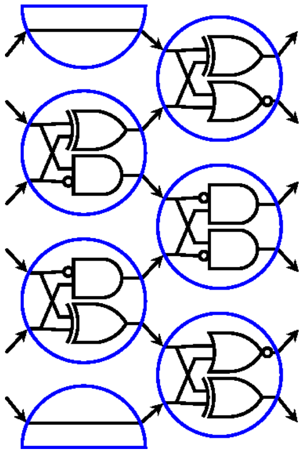
Example circuits with different gates in each row

PARITY

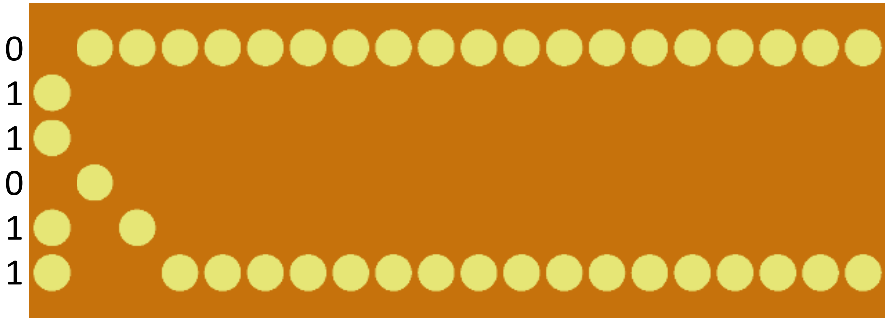




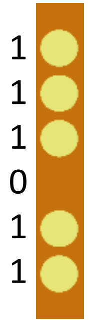
MULTIPLEOF3



$011011_2 = 27_{10} = 3 \cdot 9$

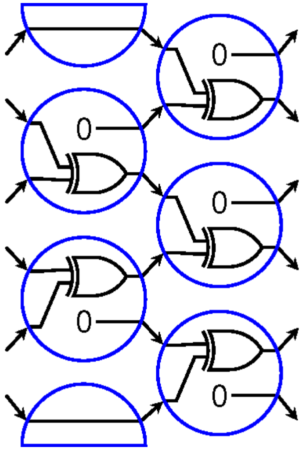


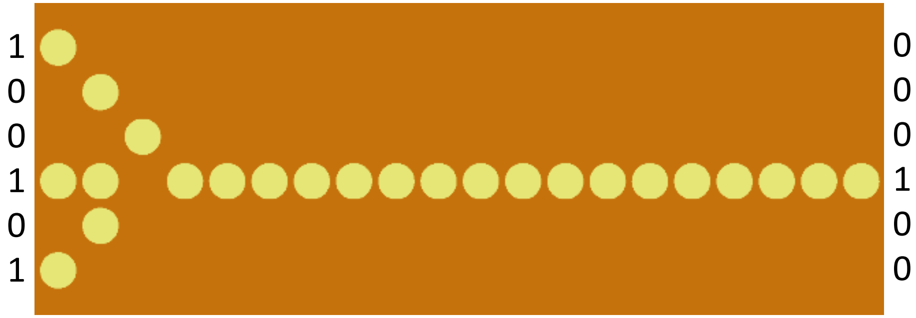
111011_2



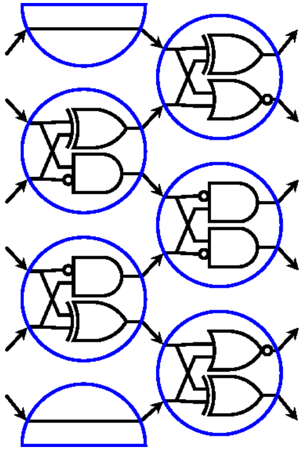
Example circuits with different gates in each row

PARITY

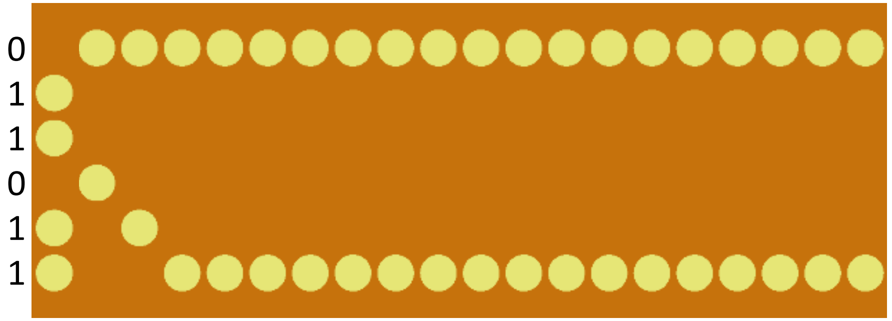




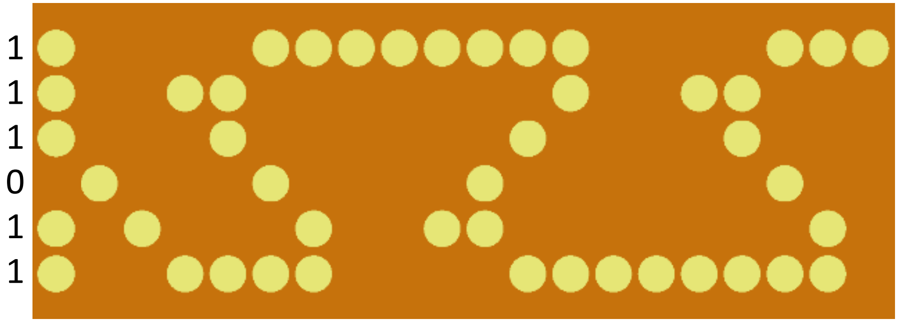
MULTIPLEOF3



$011011_2 = 27_{10} = 3 \cdot 9$

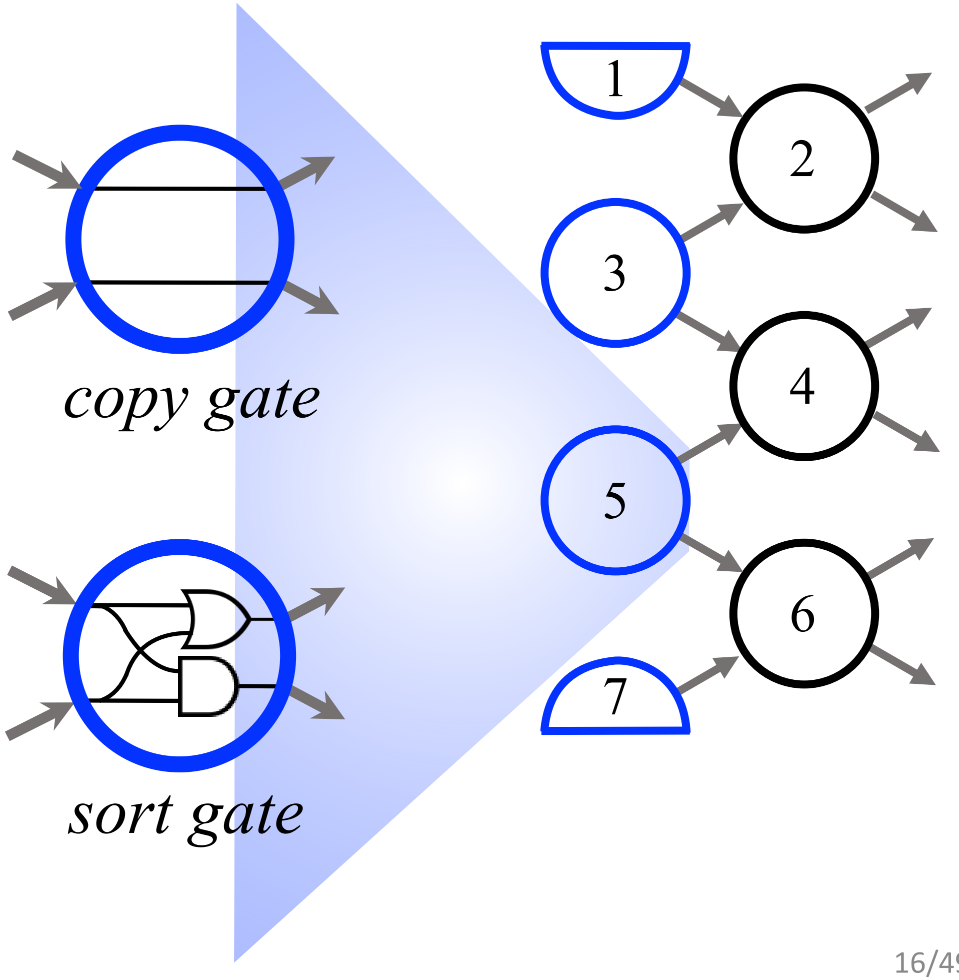


$111011_2 = 59_{10} = 3 \cdot 19 + 2$

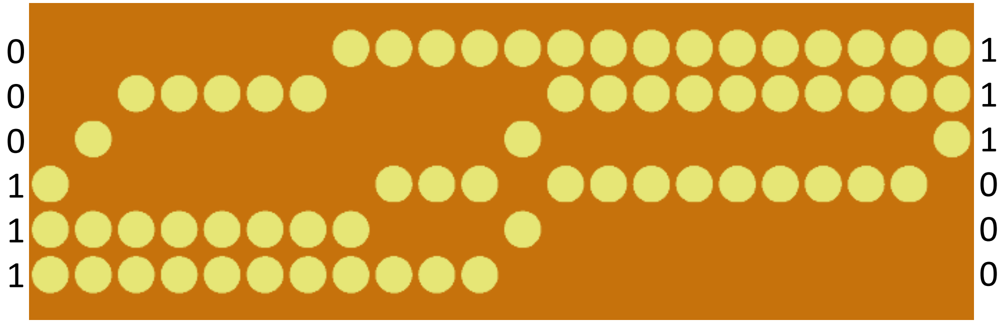


Randomization: "Lazy" sorting

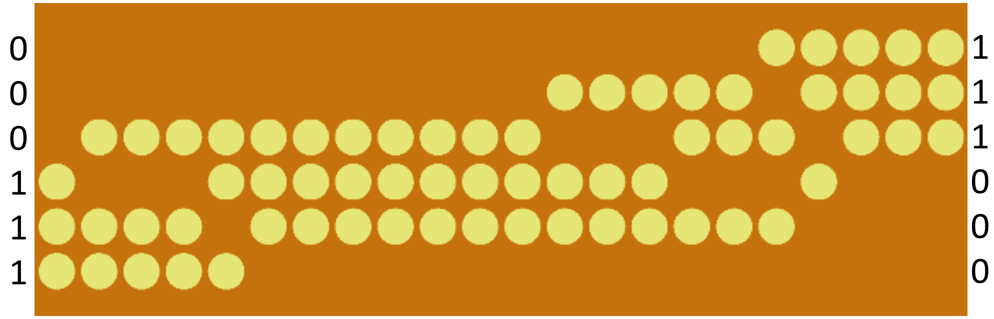
If 1 and 0 out of order, flip a coin to decide whether to swap them.

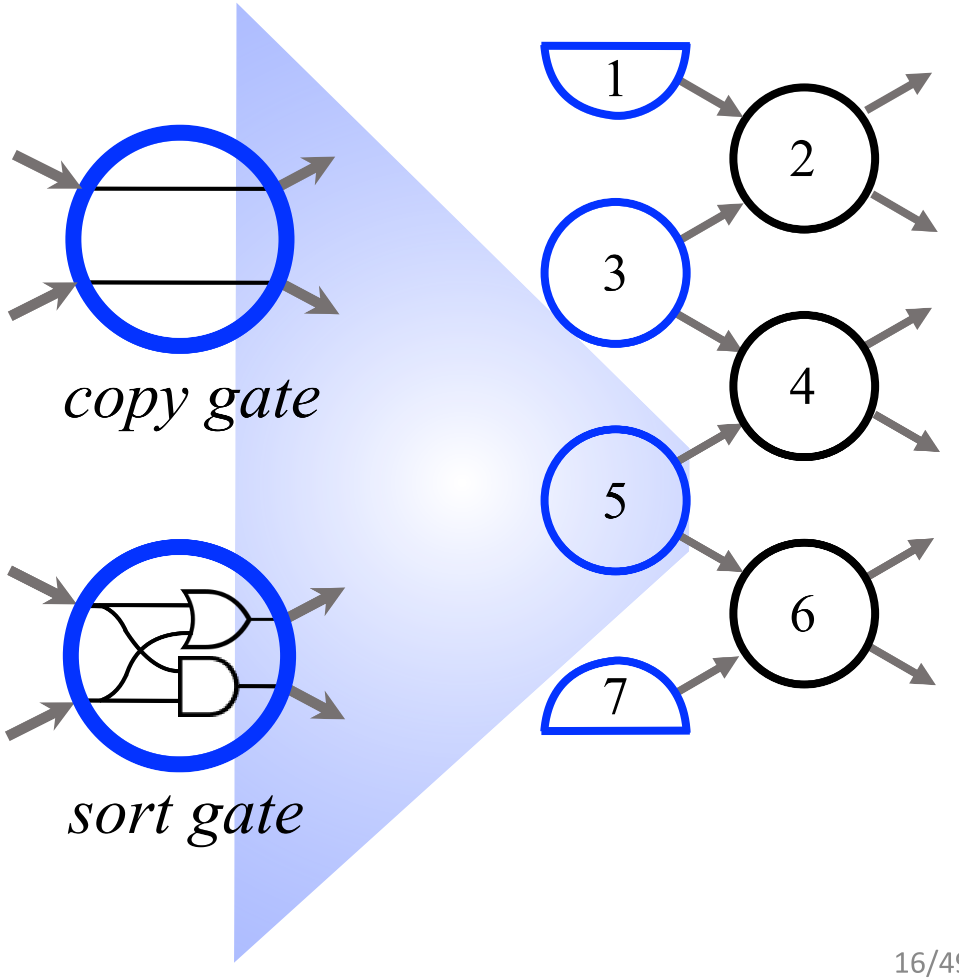


Randomization: “Lazy” sorting



If 1 and 0 out of order, flip a coin to decide whether to swap them.

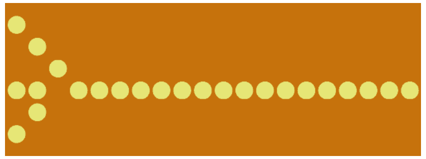




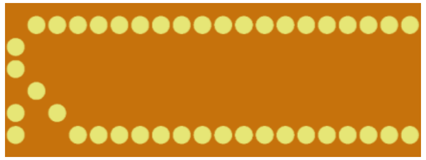
Deterministic circuits

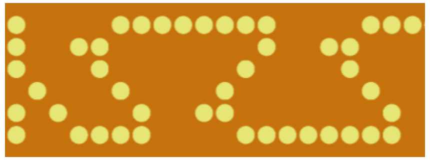
answer yes/no question

PARITY



MULTIPLEOF3

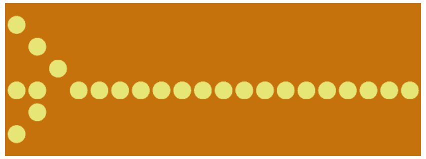




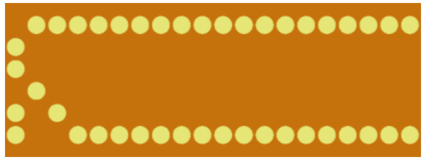
Deterministic circuits

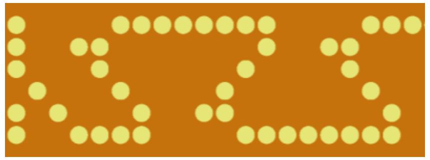
answer yes/no question

PARITY



MULTIPLEOF3





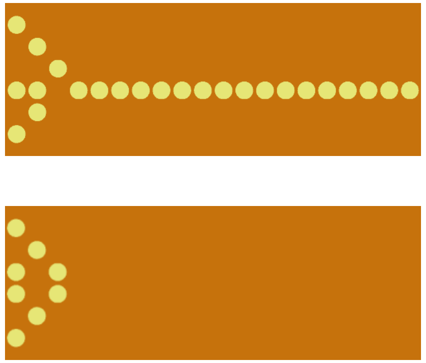
PALINDROME

yes

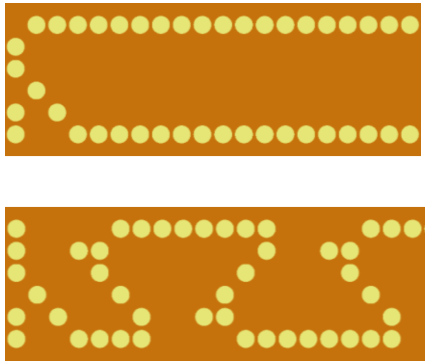
no

Deterministic circuits

PARITY

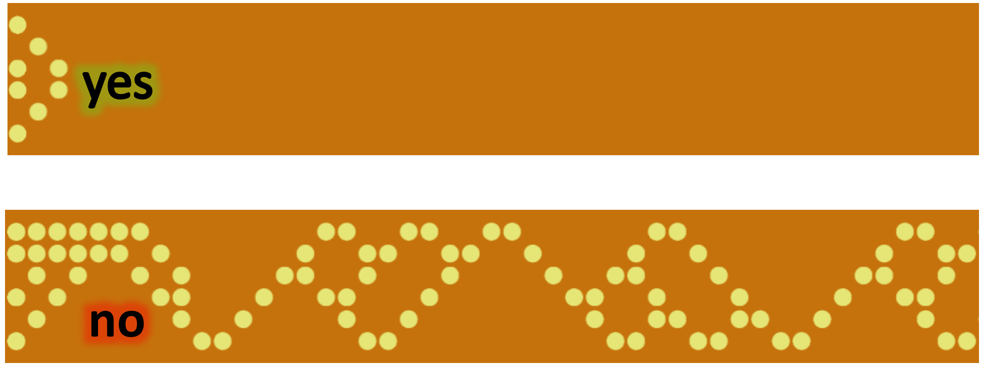


MULTIPLEOF3



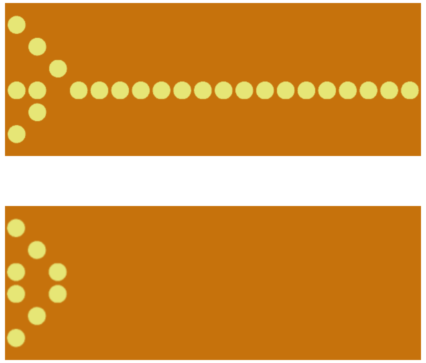
PALINDROME

answer yes/no question

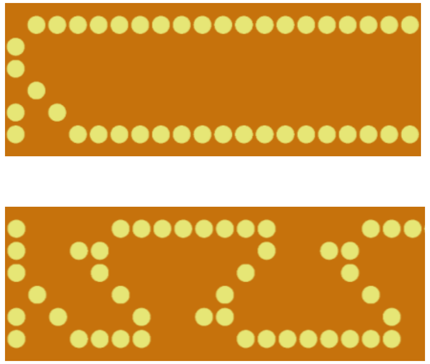


Deterministic circuits

Parity

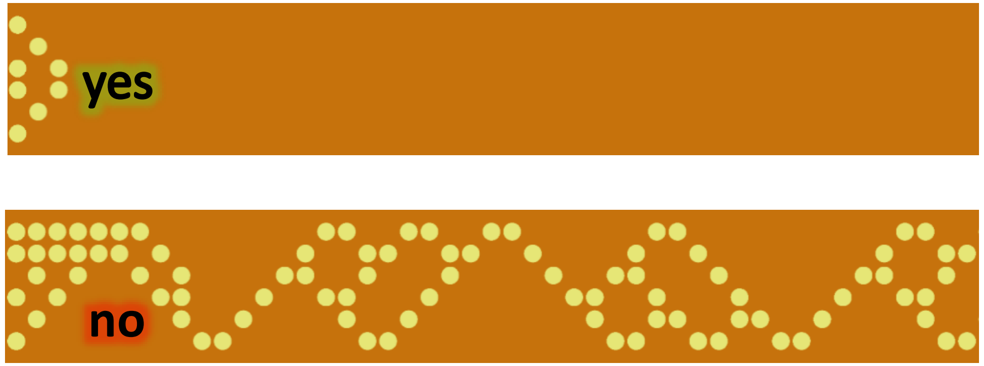


MultipleOf3



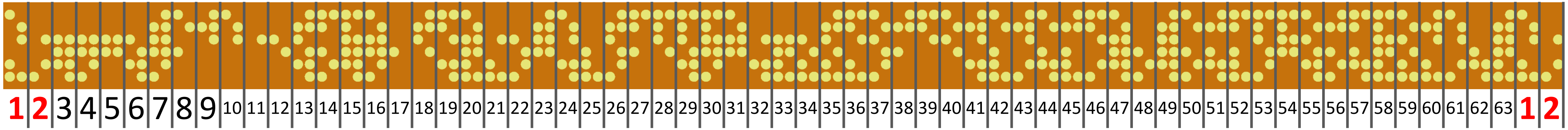
Palindrome

answer yes/no question



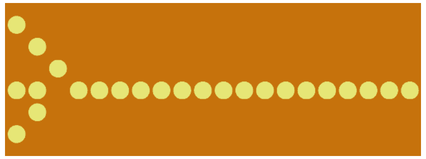
Cycle63

"count" as high as possible

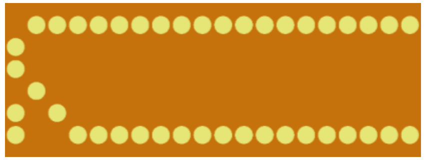


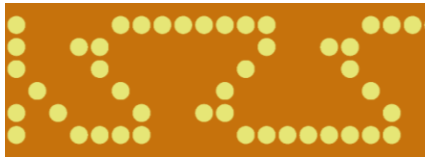
Deterministic circuits

PARITY



MULTIPLEOF3



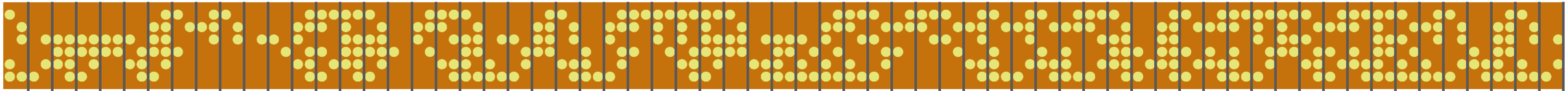


PALINDROME

answer yes/no question

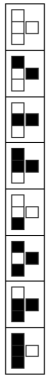
CYCLE63

"count" as high as possible



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 1 2

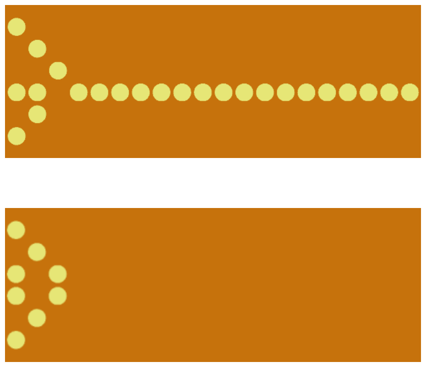
RULE110



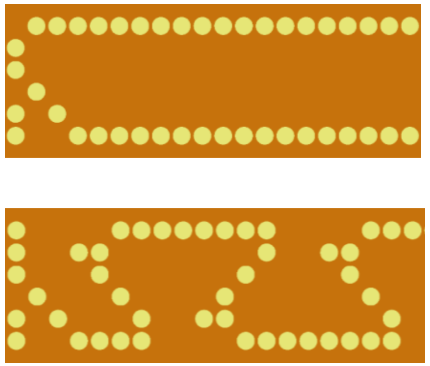
simulate cellular automata

Deterministic circuits

PARITY

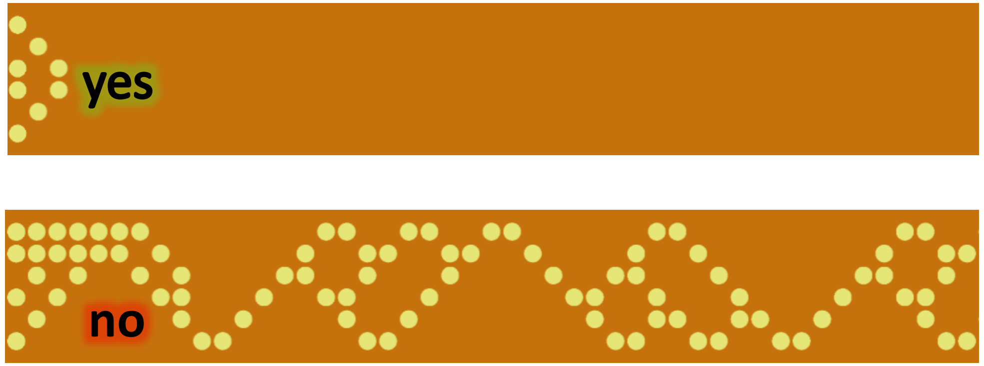


MULTIPLEOF3



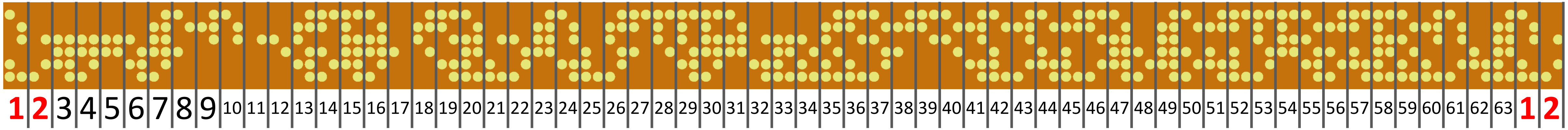
PALINDROME

answer yes/no question



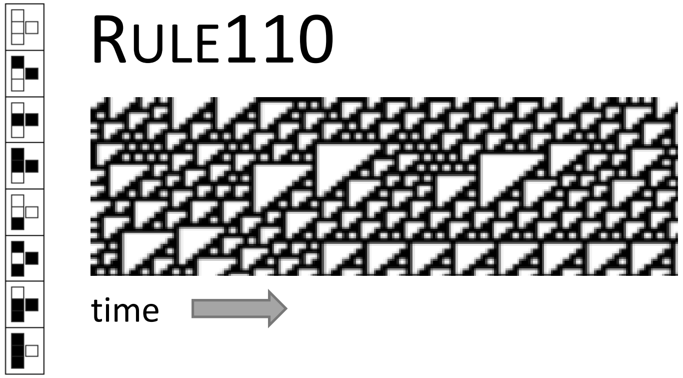
CYCLE63

"count" as high as possible



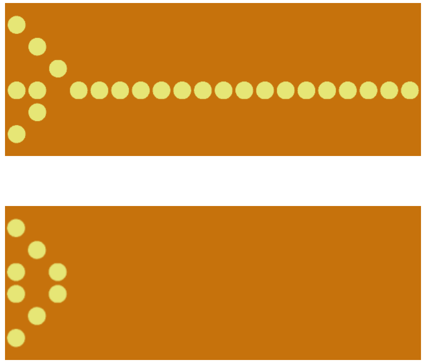
RULE110

simulate cellular automata

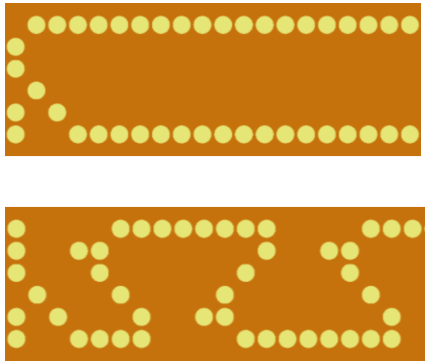


Deterministic circuits

PARITY

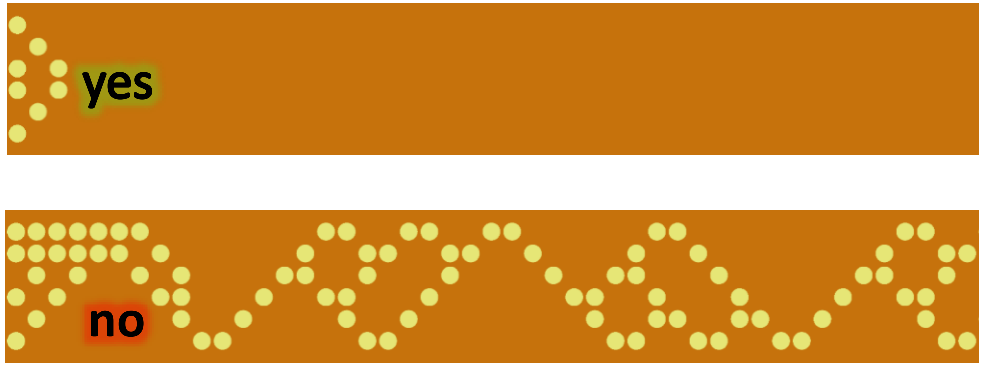


MULTIPLEOF3



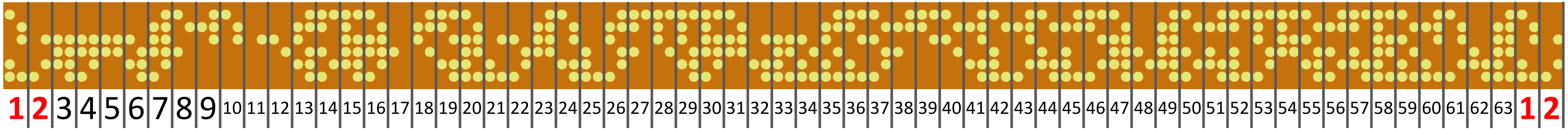
PALINDROME

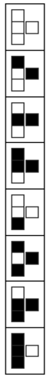
answer yes/no question



CYCLE63

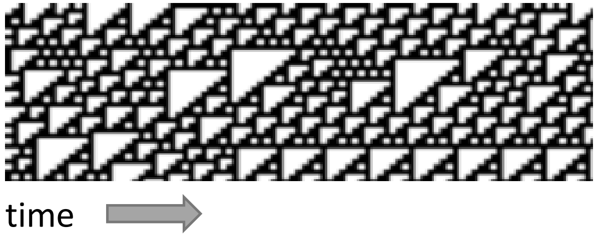
“count” as high as possible

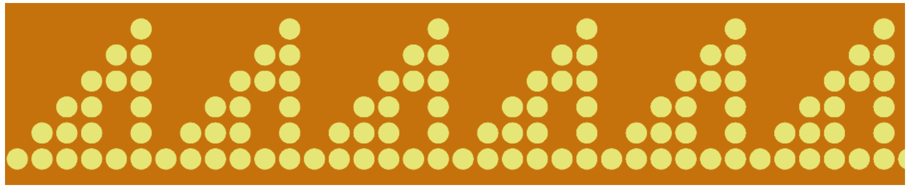




RULE110

simulate cellular automata





Theorem: Rule 110 can efficiently execute <u>any</u> algorithm.

[Cook, <u>Complex Systems</u> 2004]

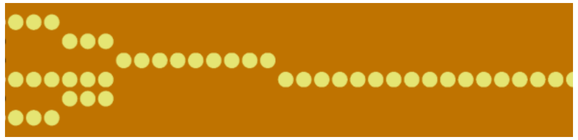
[Neary, Woods, <u>ICALP</u> 2006]

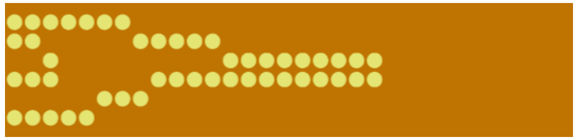
Randomized circuits

LAZYPARITY

Randomized circuits

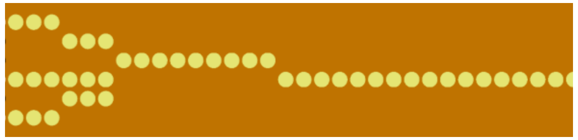
LAZYPARITY

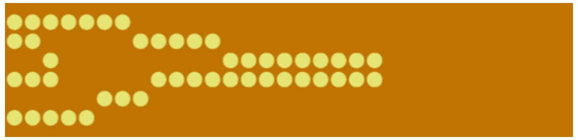




Randomized circuits

LAZYPARITY

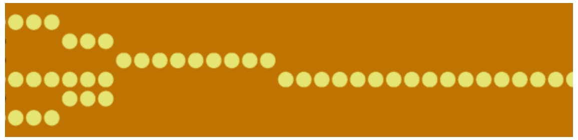


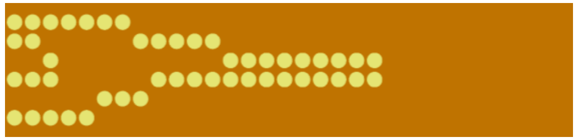


RANDOMWALKINGBIT

Randomized circuits

LazyParity





RandomWalkingBit

DiamondsAreForever

Randomized circuits

LazyParity

RandomWalkingBit

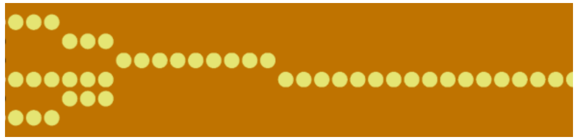
DiamondsAreForever

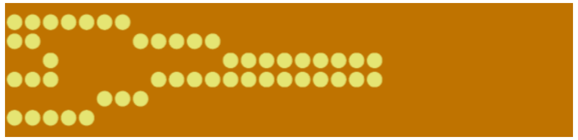
FairCoin

use biased coin to
simulate unbiased coin

Randomized circuits

LazyParity





RandomWalkingBit

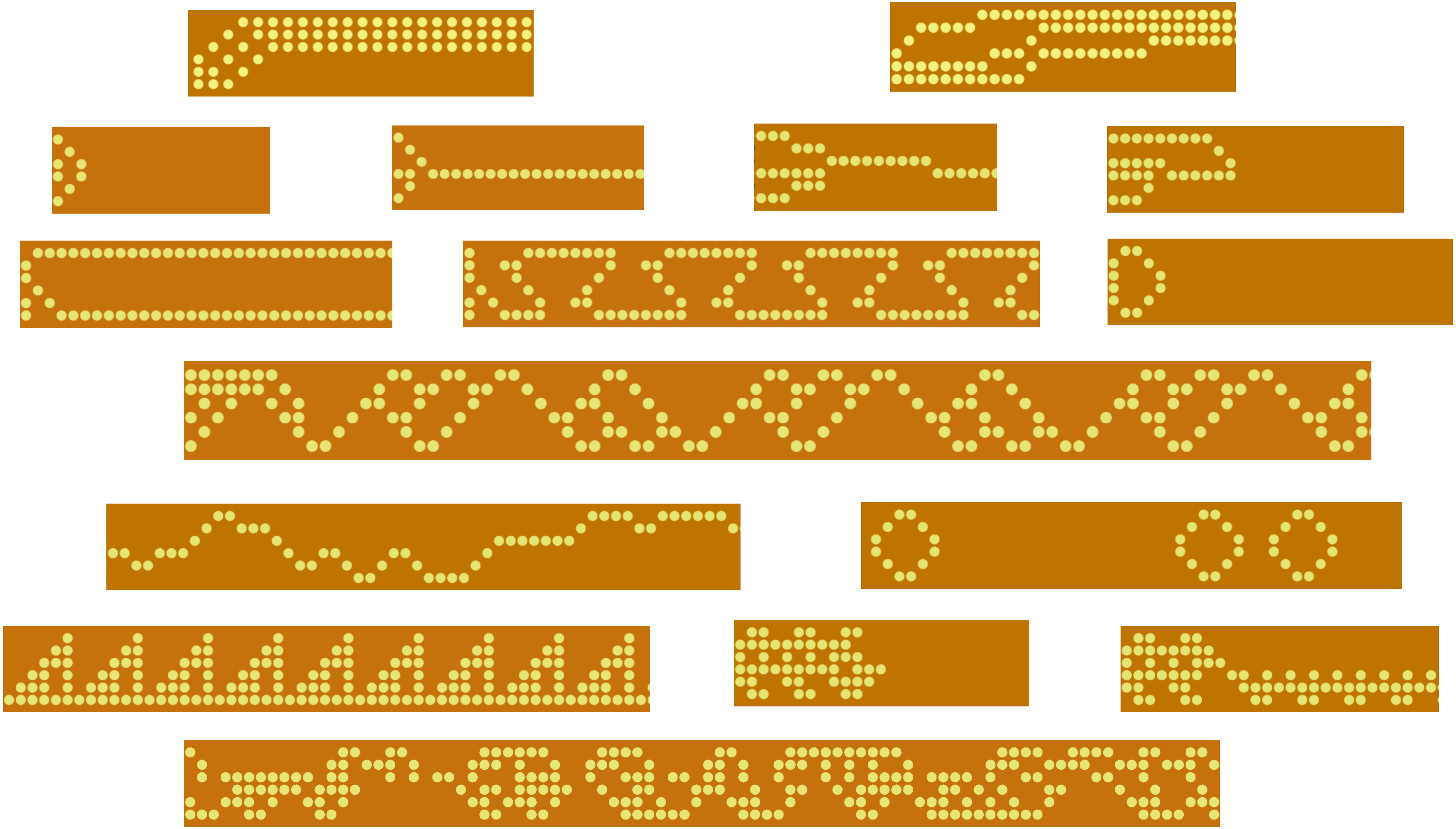
DiamondsAreForever

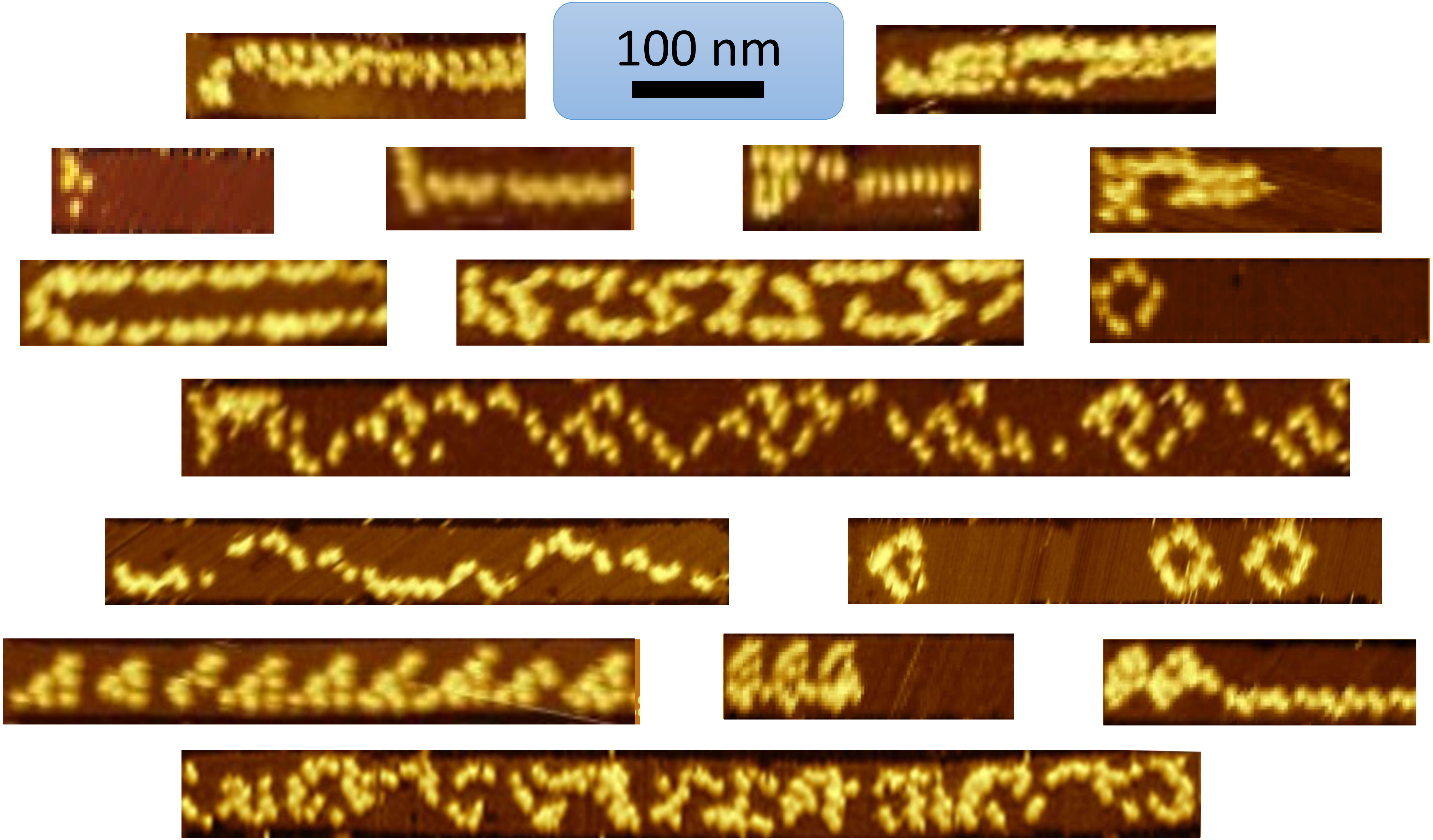
FairCoin

use biased coin to
simulate unbiased coin

Pr () = Pr () = ½

for **any** (positive) probabilities for the randomized gate



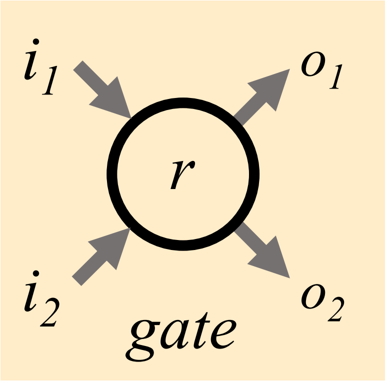

100 nm

Hierarchy of abstractions

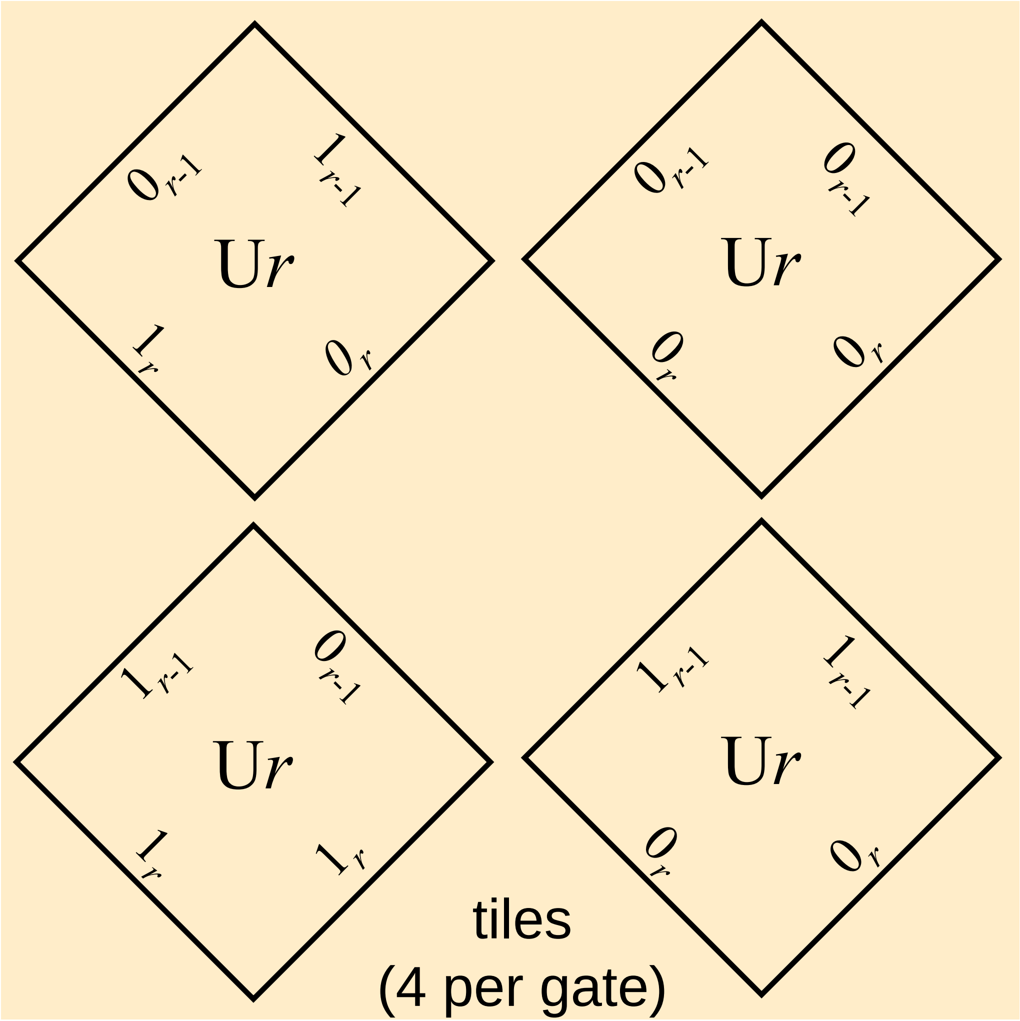
Bits:	Boolean circuits compute
Tiles:	**Tile self-assembly implements circuits**
DNA:	DNA strands implement tiles

Gates → Tiles

1. Compile gates to tiles

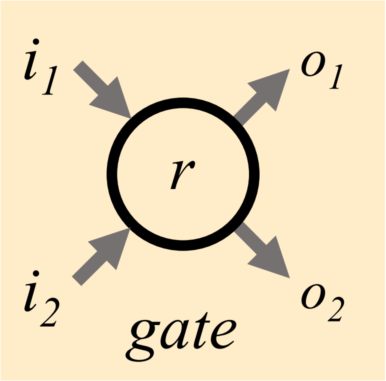


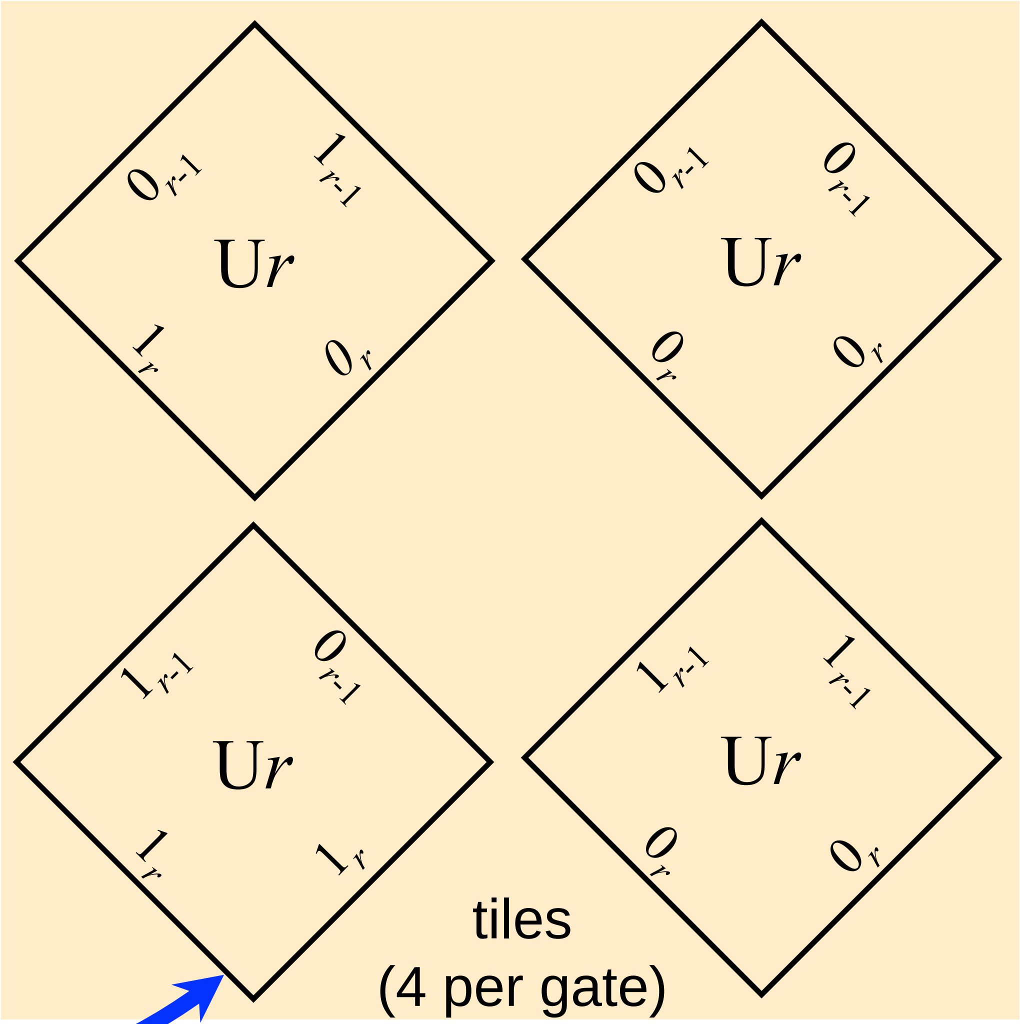
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Gates → Tiles

1. Compile gates to tiles





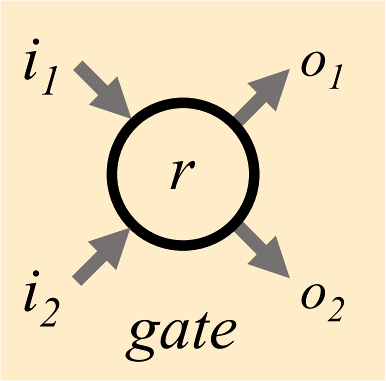
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

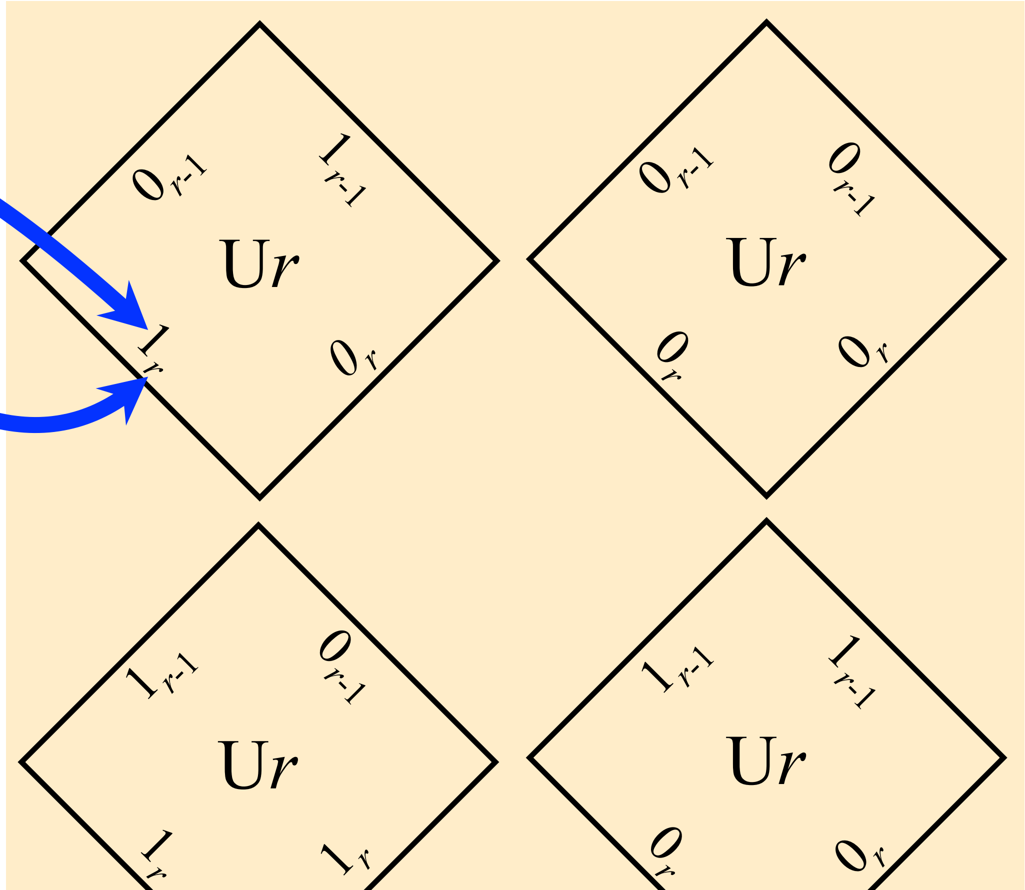
Each **entry** of a gate's truth table is encoded by a **tile**

Gates → Tiles

1. Compile gates to tiles

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



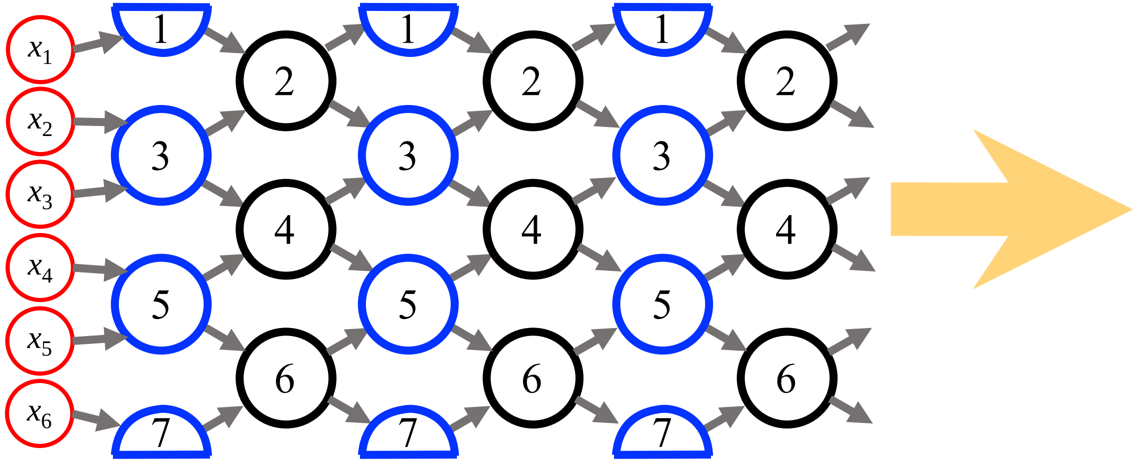


tiles
(4 per gate)

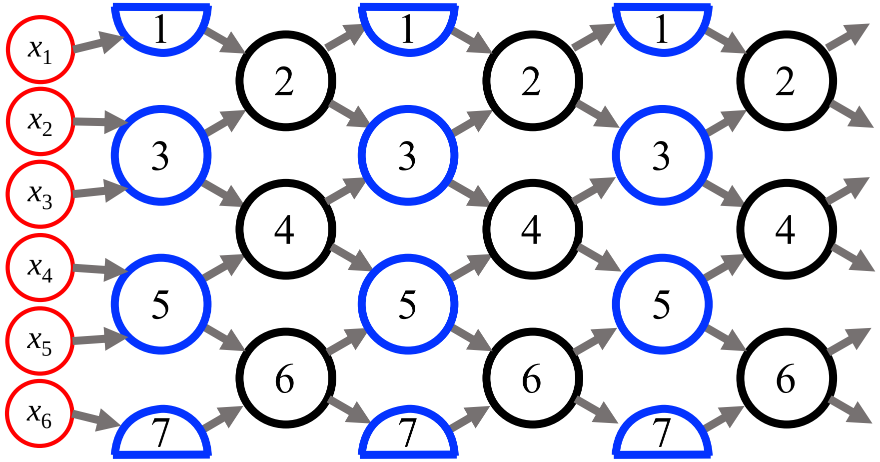
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

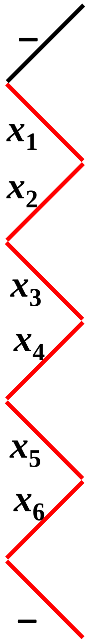
Each **entry** of a gate's truth table is encoded by a **tile**

How tiles compute while growing (algorithmic self-assembly)

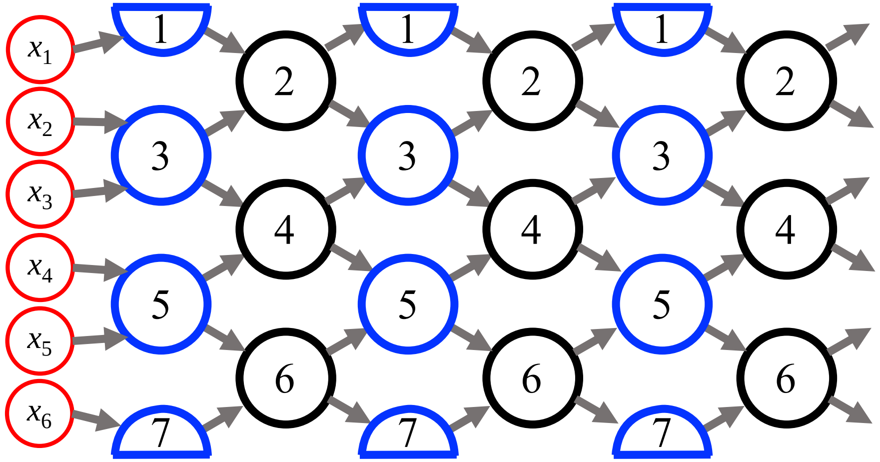


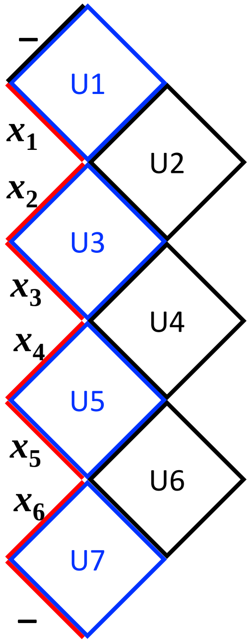
How tiles compute while growing (algorithmic self-assembly)



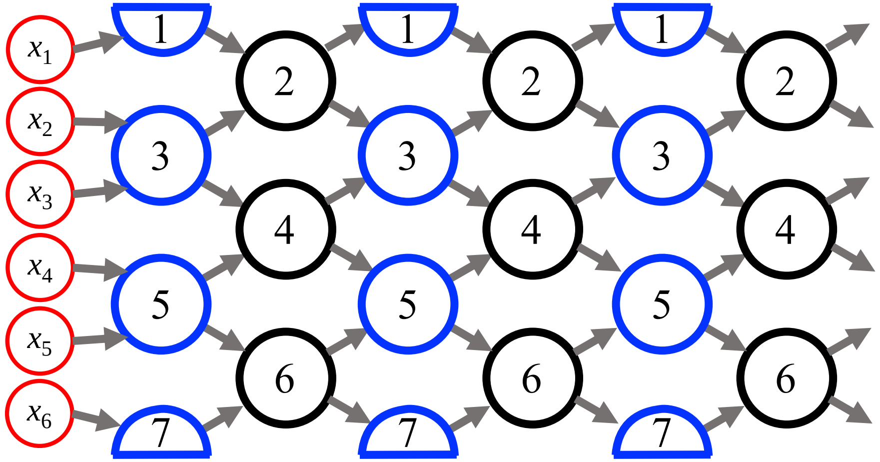


How tiles compute while growing (algorithmic self-assembly)

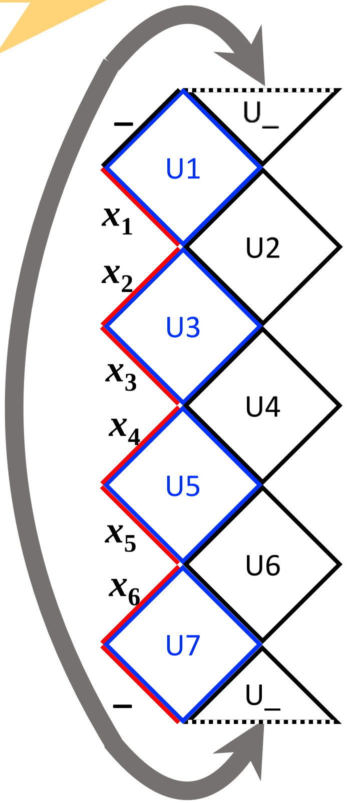




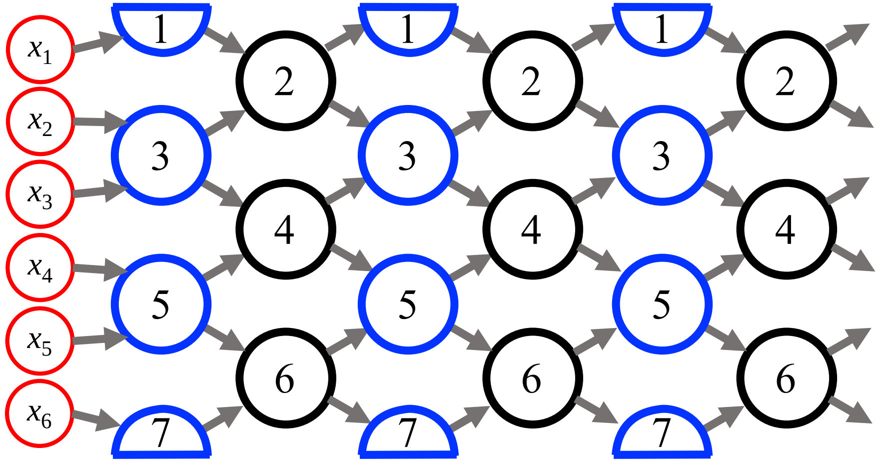
How tiles compute while growing (algorithmic self-assembly)

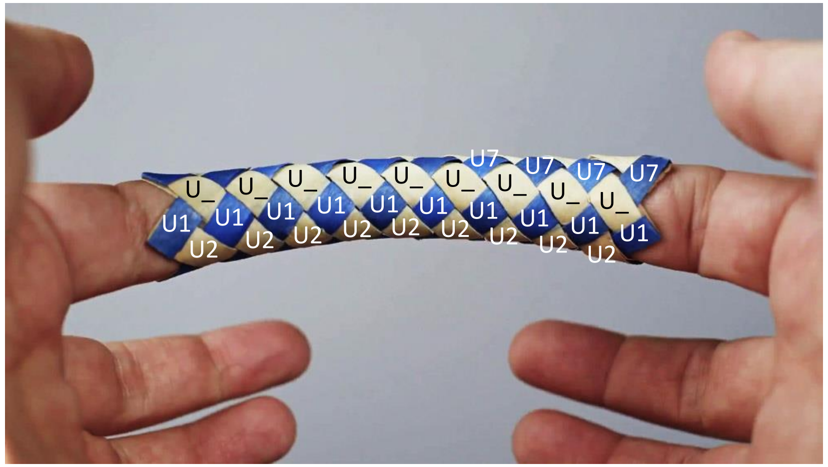


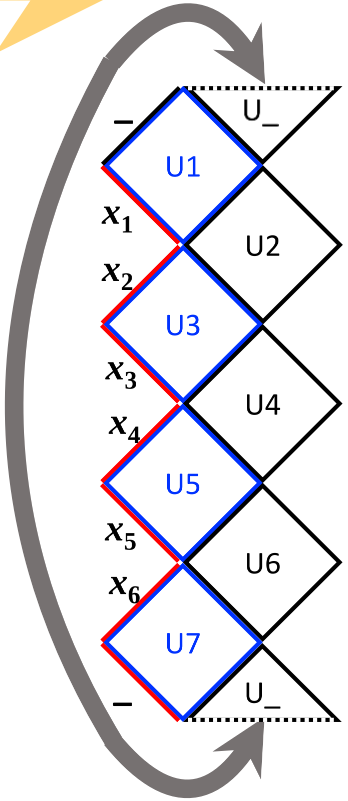
“data-free” tile wraps top to bottom to form a tube



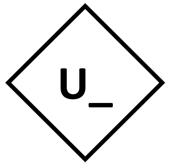
How tiles compute while growing (algorithmic self-assembly)



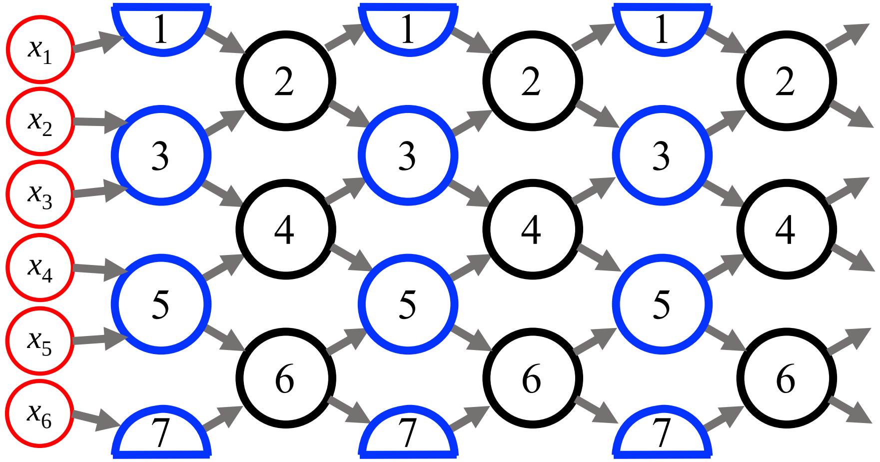


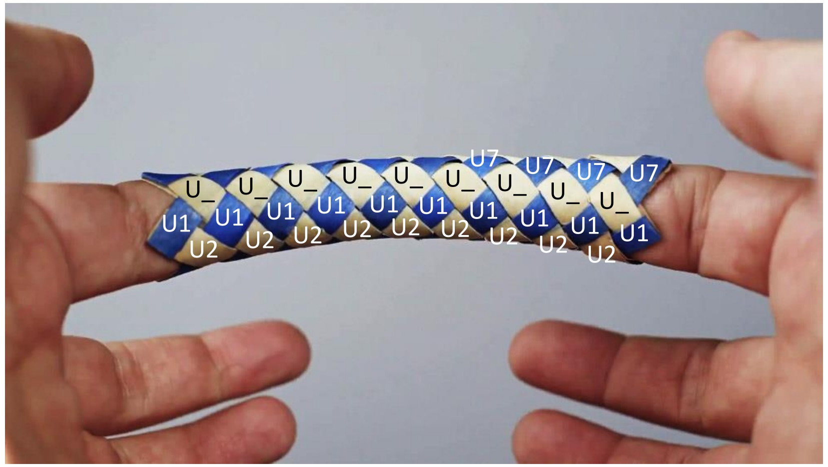


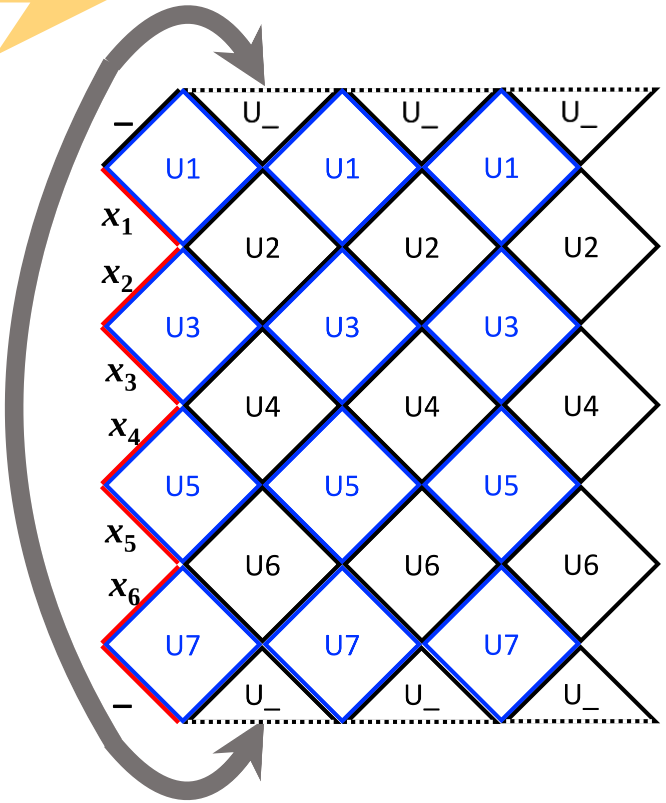
“data-free” tile wraps top to bottom to form a tube



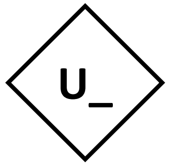
How tiles compute while growing (algorithmic self-assembly)



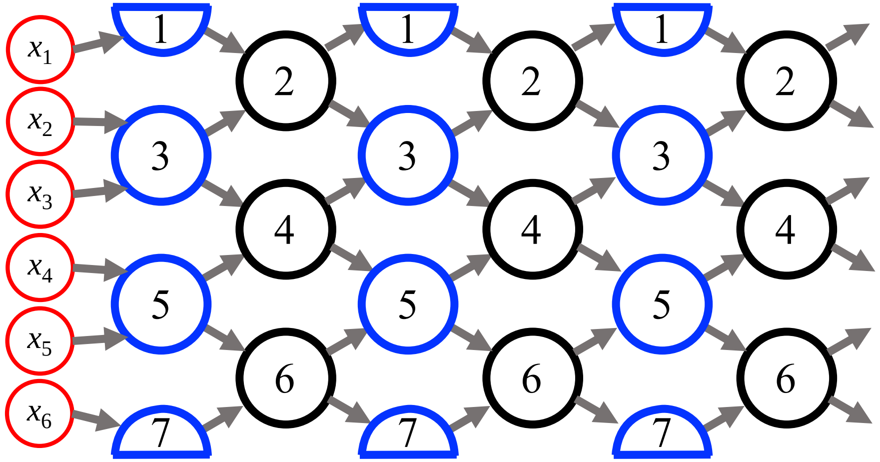


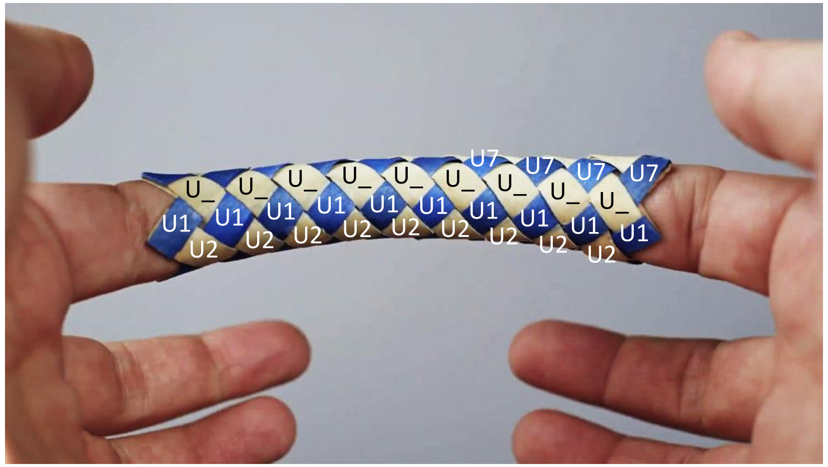


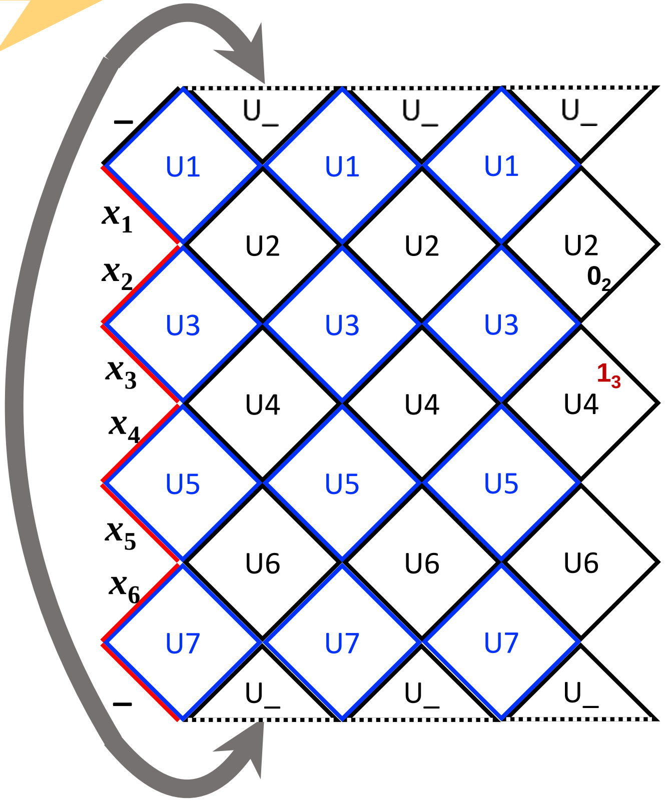
“data-free” tile wraps top to bottom to form a tube



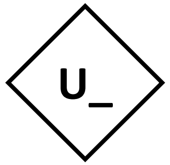
How tiles compute while growing (algorithmic self-assembly)



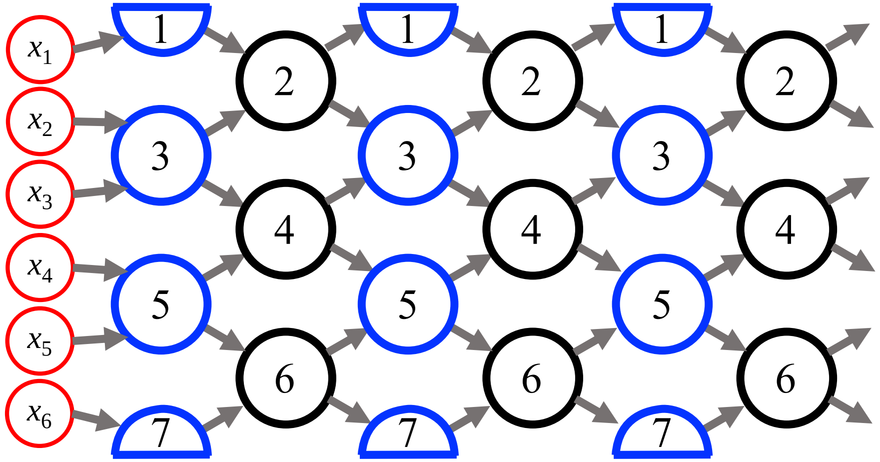


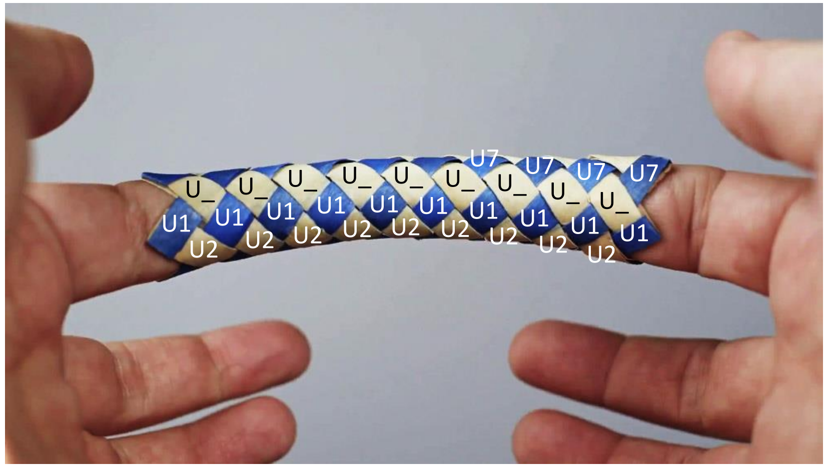


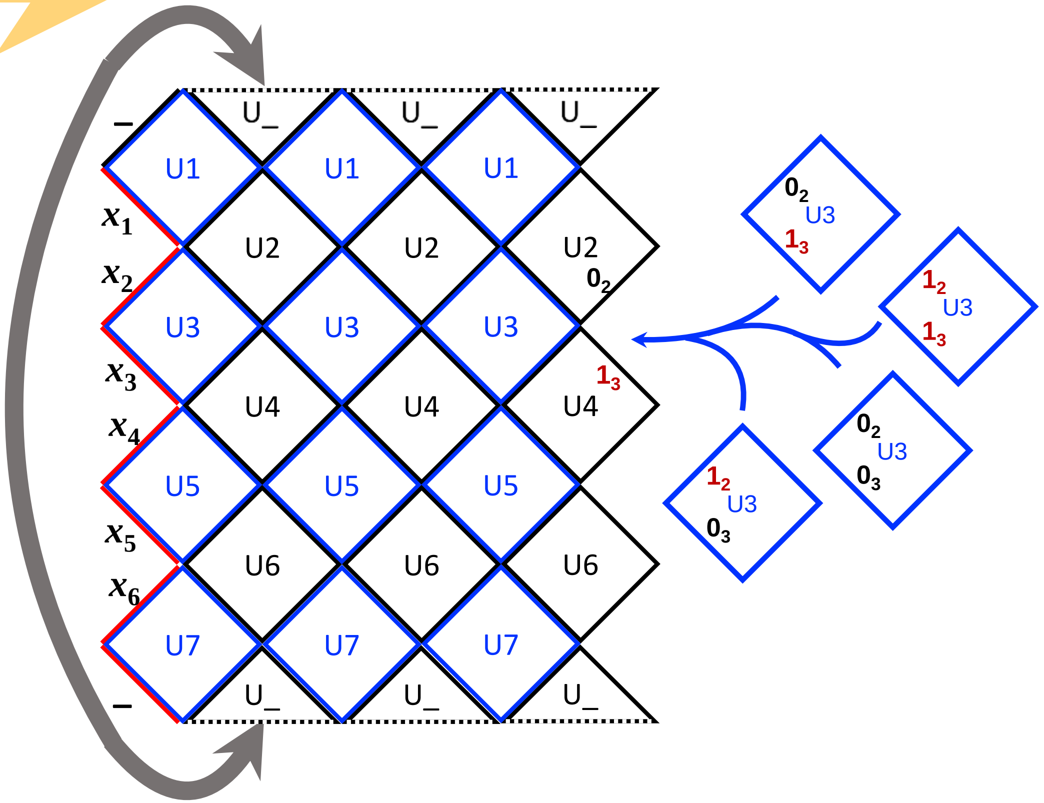
"data-free" tile wraps top to bottom to form a tube



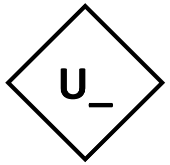
How tiles compute while growing (algorithmic self-assembly)



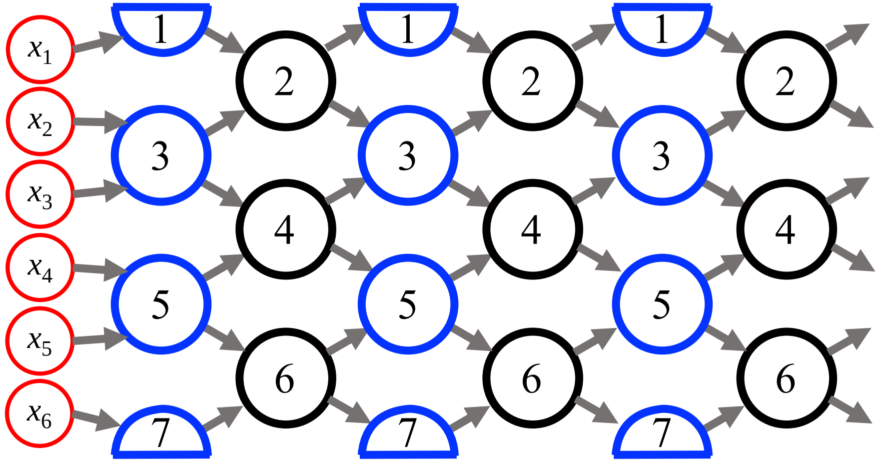


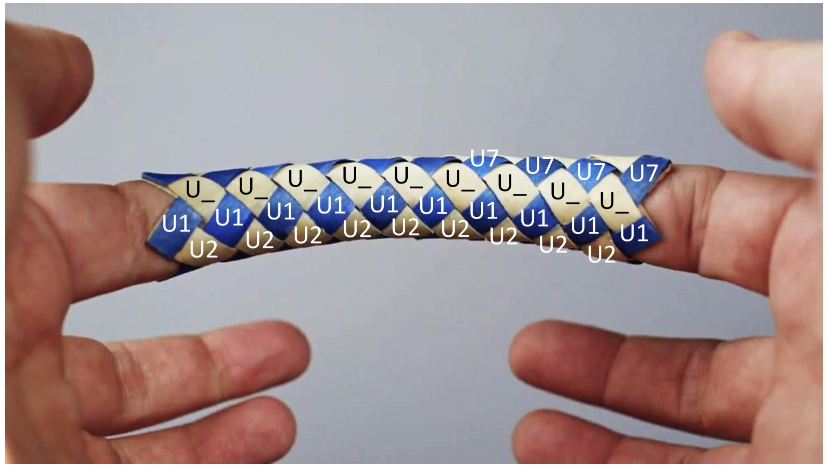


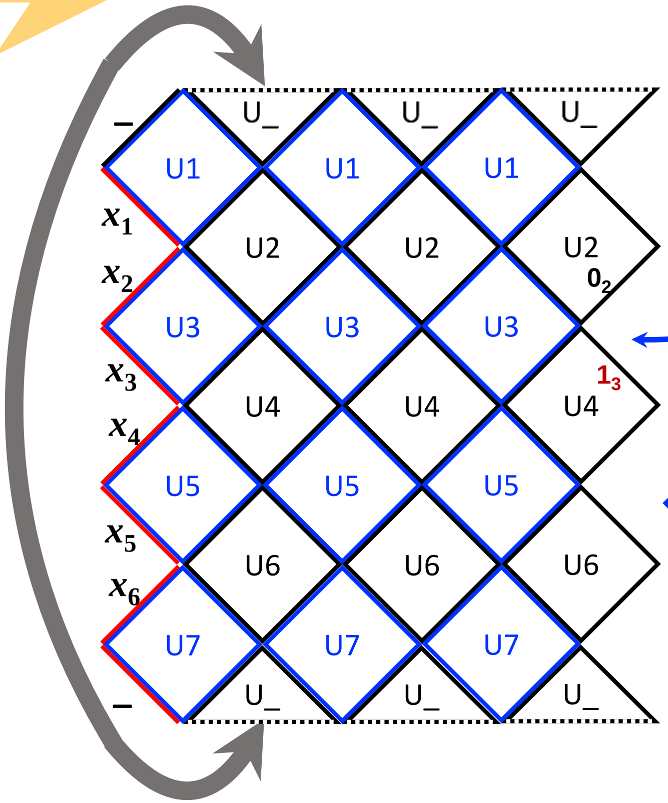
“data-free” tile wraps top to bottom to form a tube

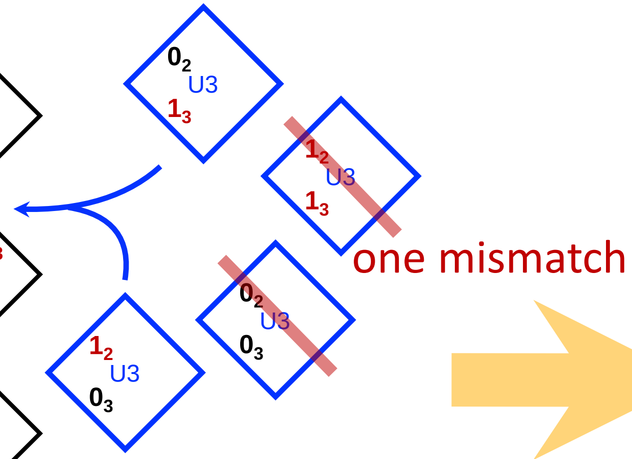


How tiles compute while growing (algorithmic self-assembly)

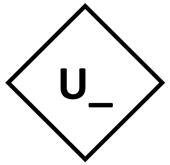




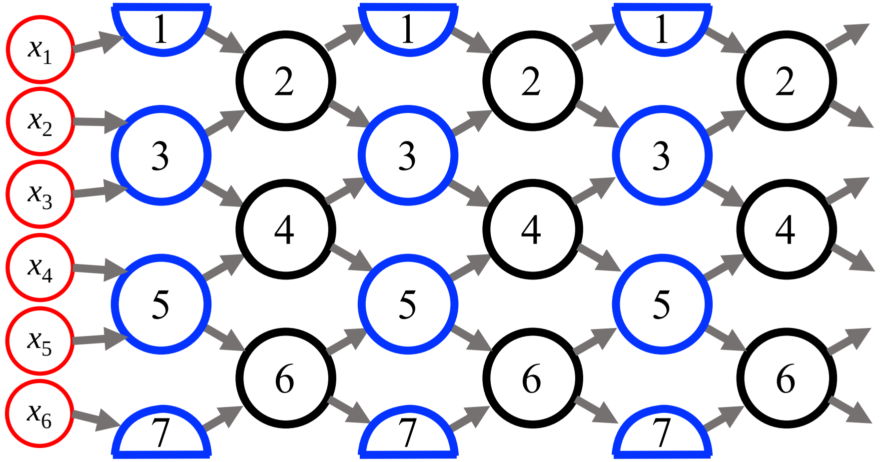


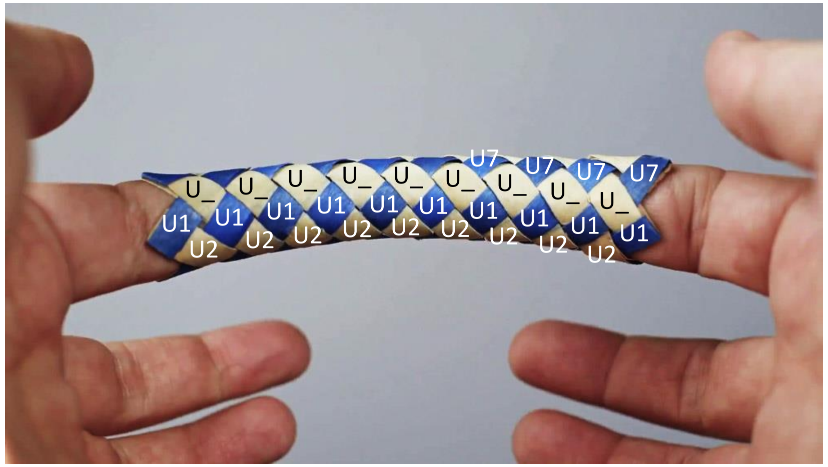


"data-free" tile wraps top to bottom to form a tube

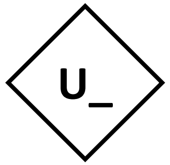


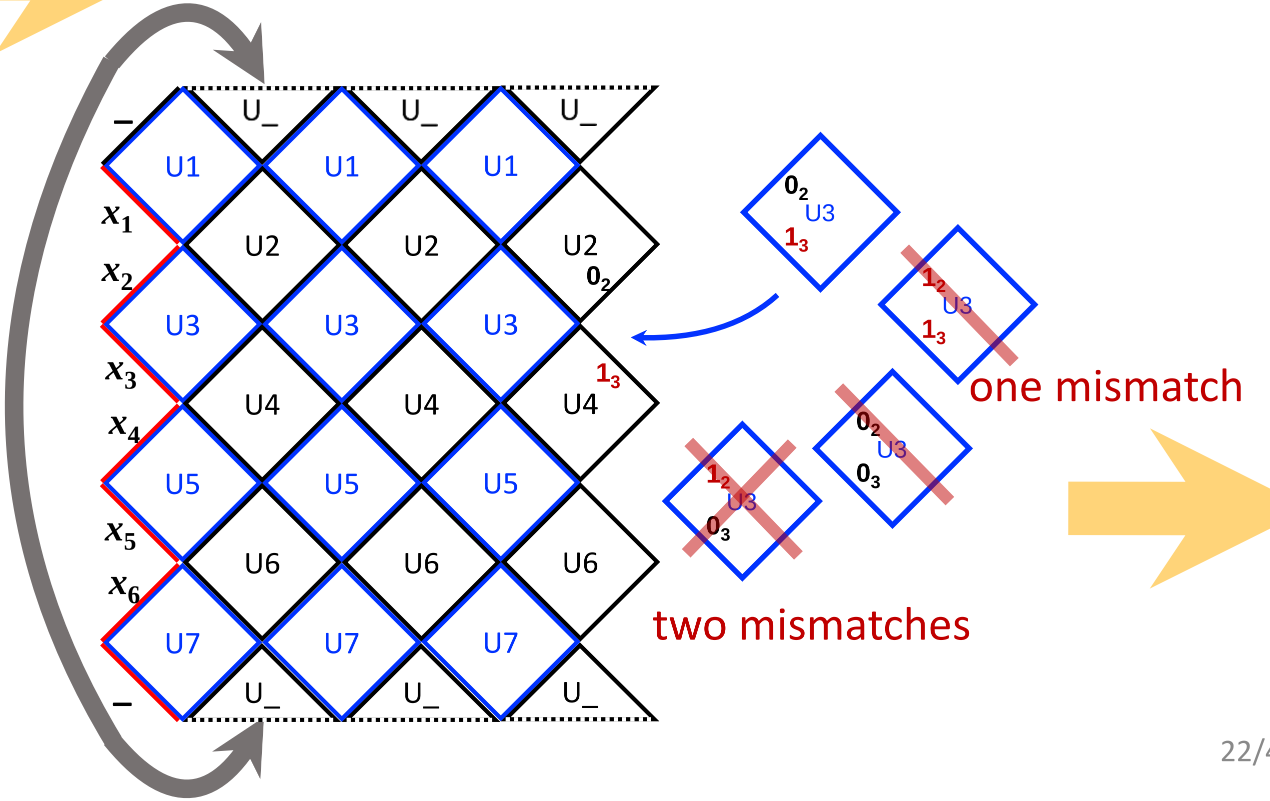
How tiles compute while growing (algorithmic self-assembly)



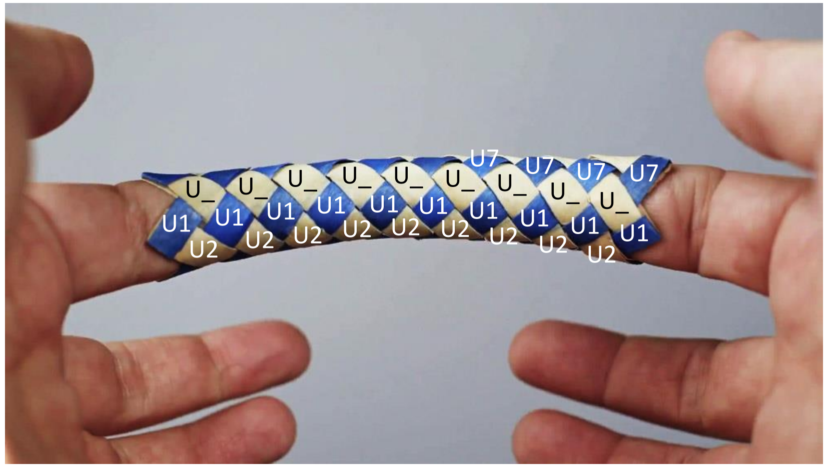


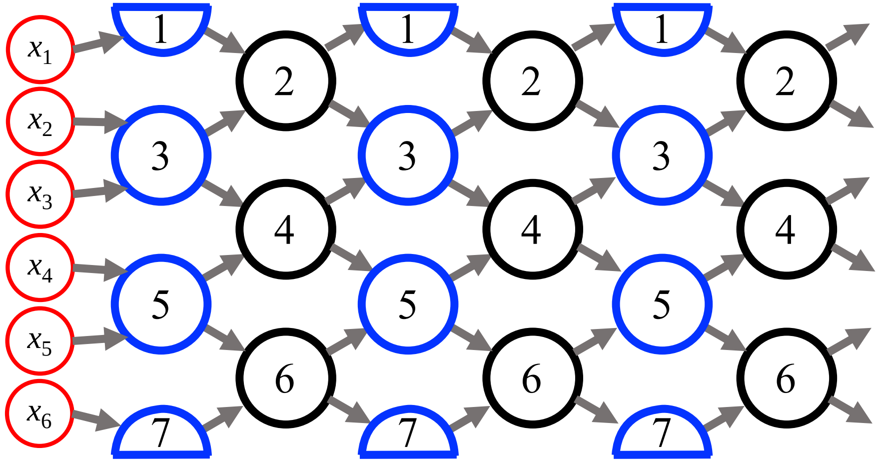
“data-free” tile wraps top to bottom to form a tube

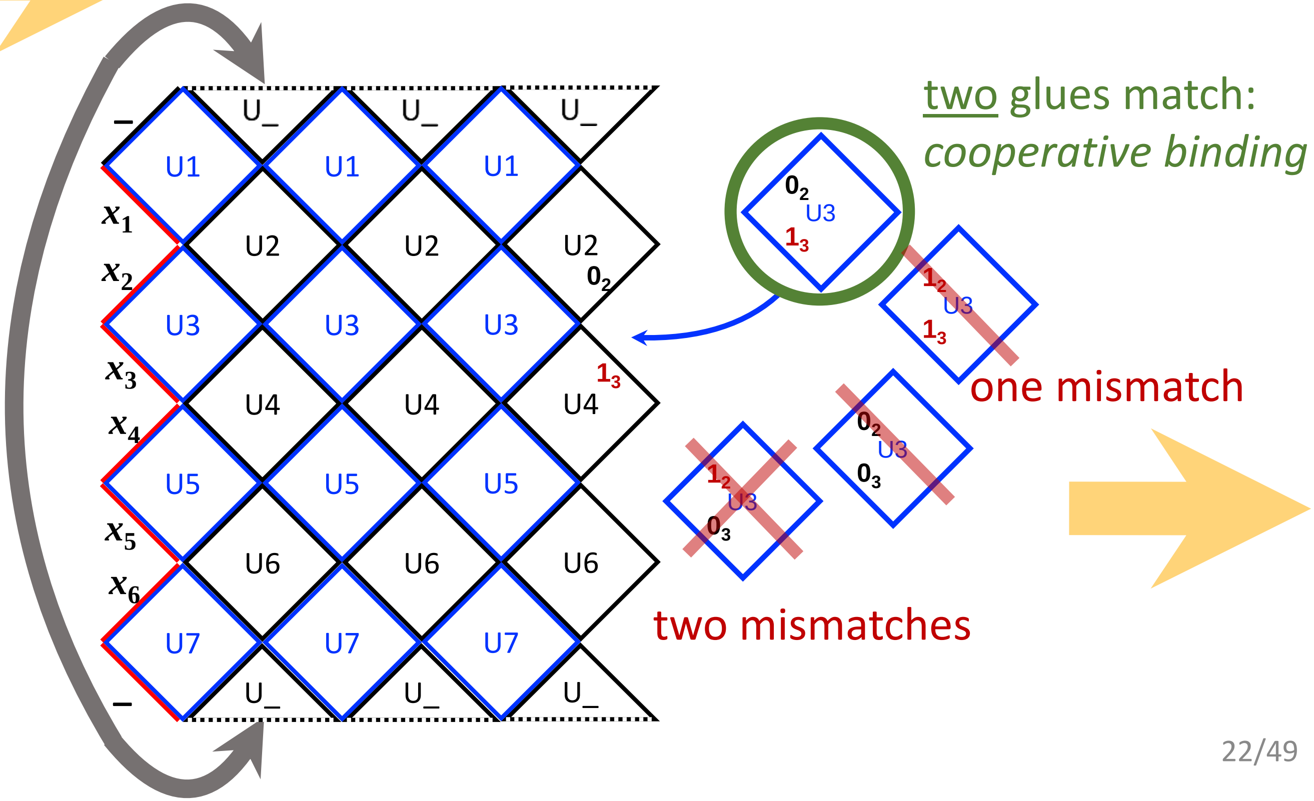




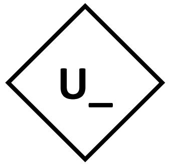
How tiles compute while growing (algorithmic self-assembly)



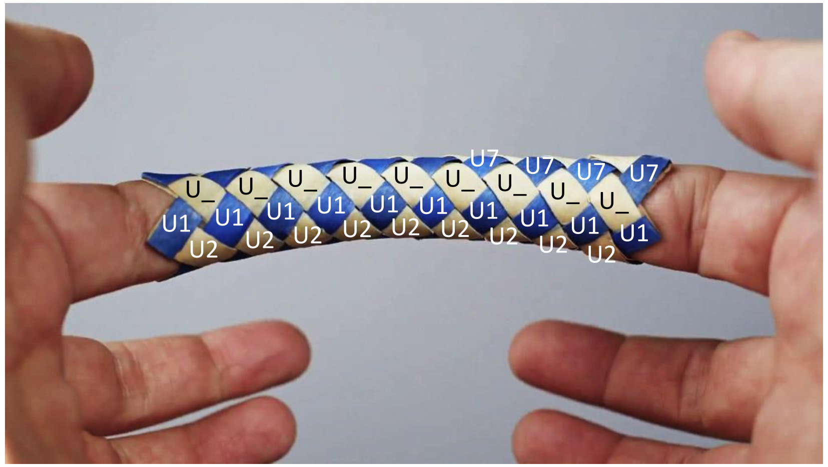


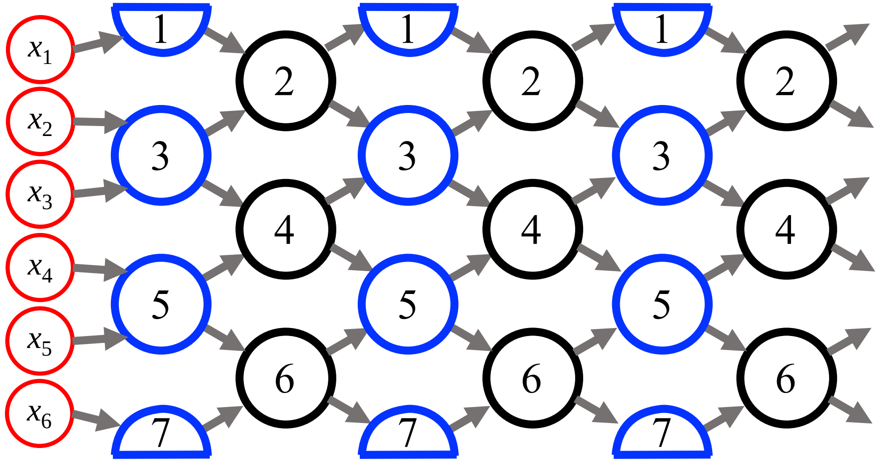


"data-free" tile wraps top to bottom to form a tube

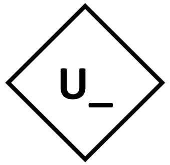


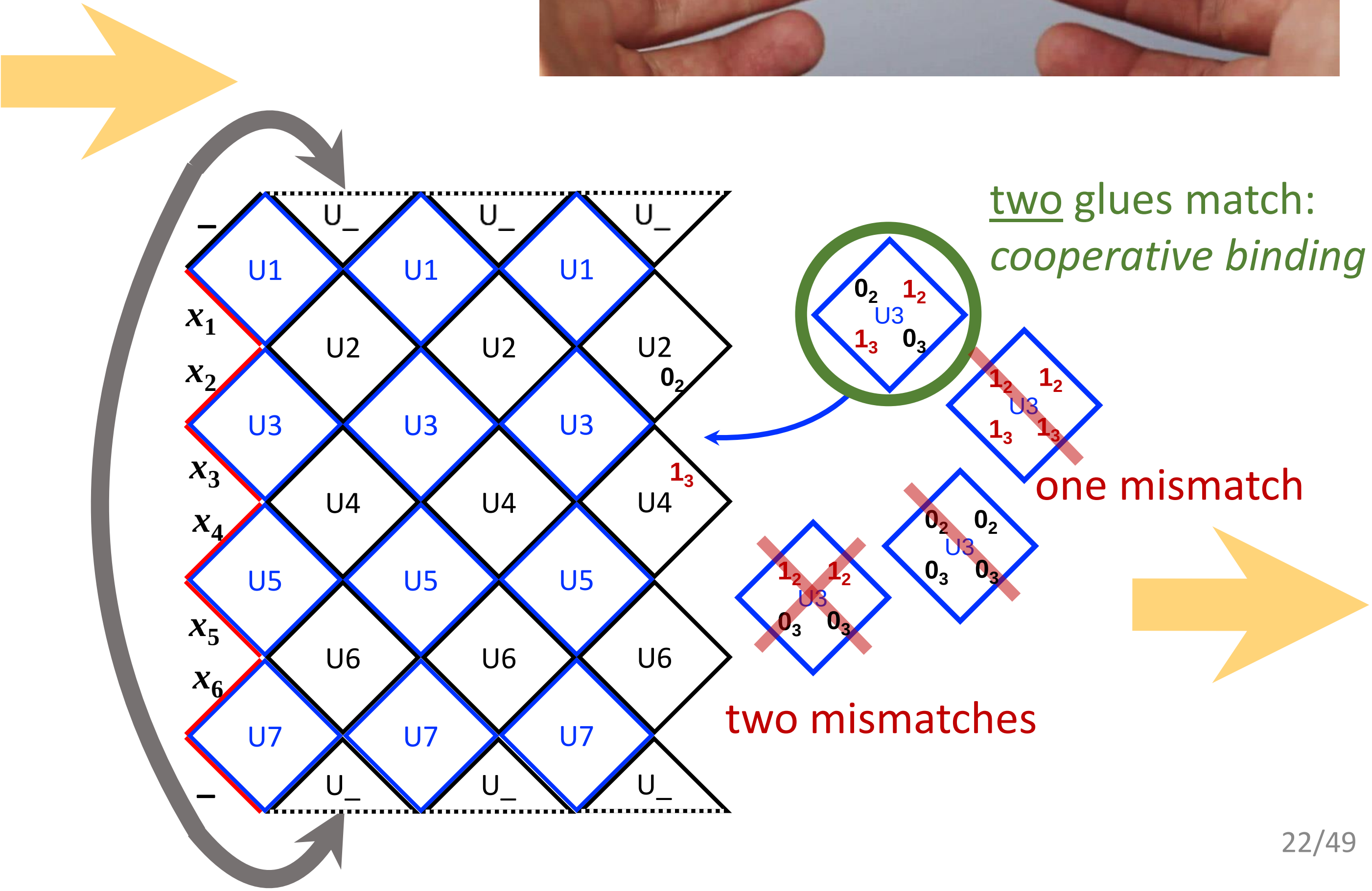
How tiles compute while growing (algorithmic self-assembly)





"data-free" tile wraps top to bottom to form a tube





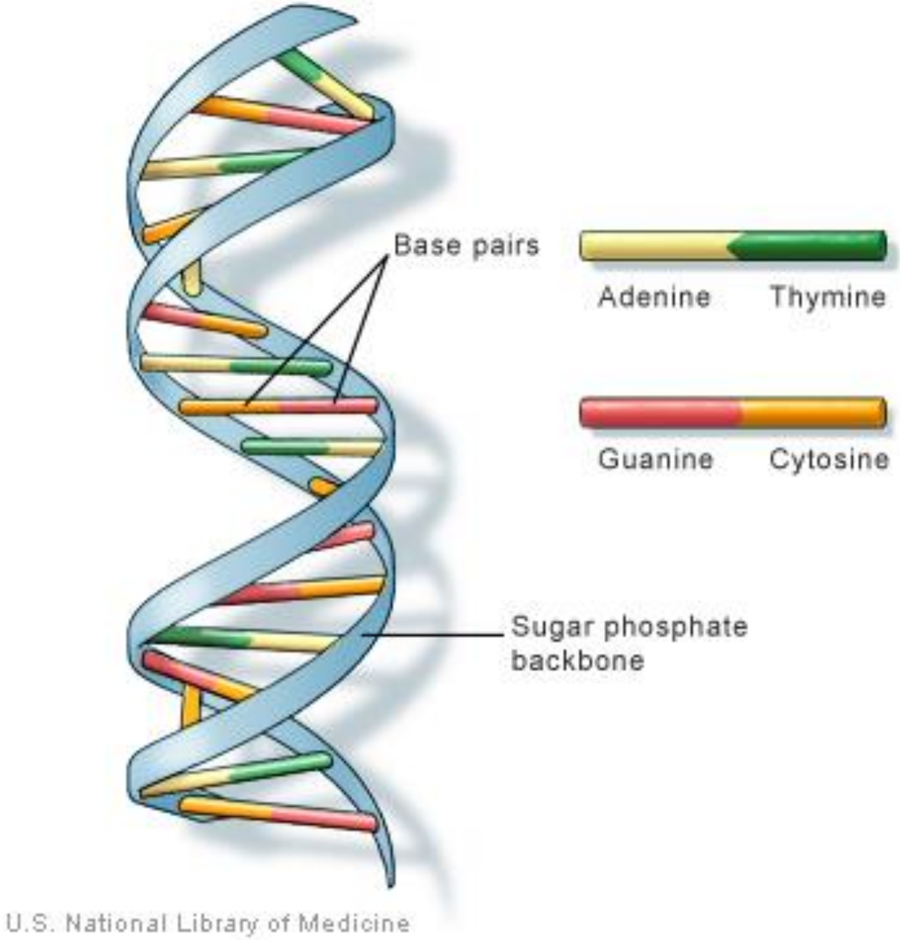
Hierarchy of abstractions

Bits:	Boolean circuits compute
Tiles:	Tile self-assembly implements circuits
DNA:	**DNA strands implement tiles**

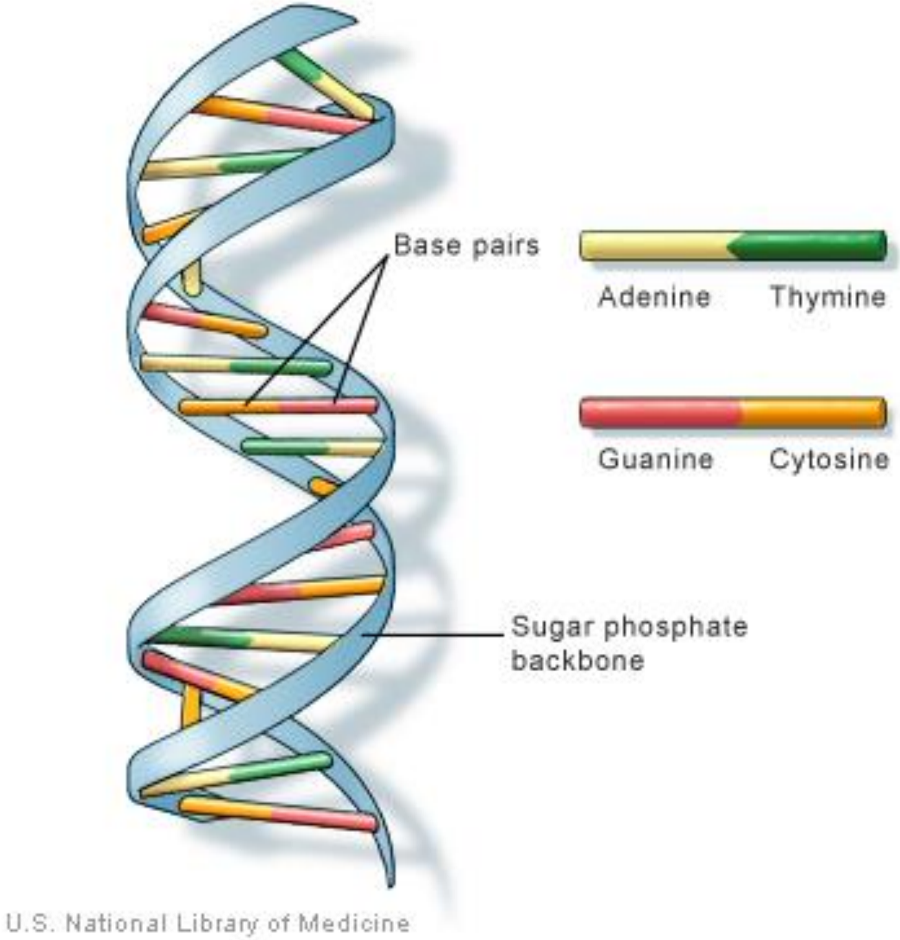
Structural DNA nanotechnology

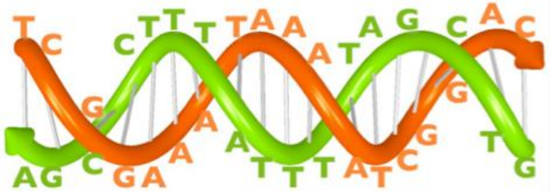
a.k.a. DNA carpentry

DNA as a building material



DNA as a building material



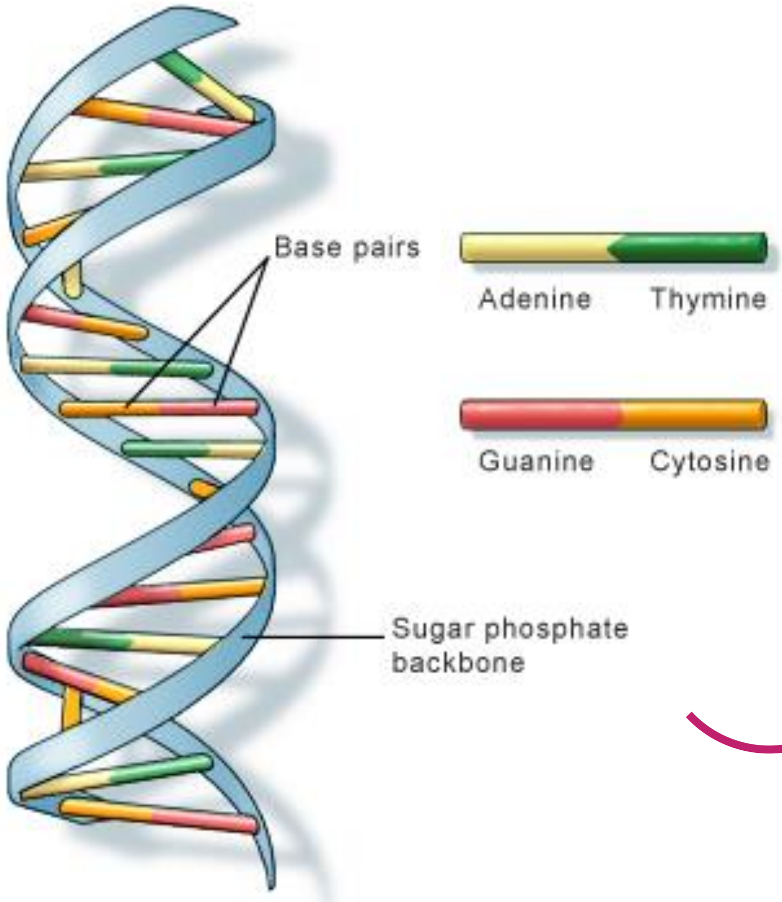


=

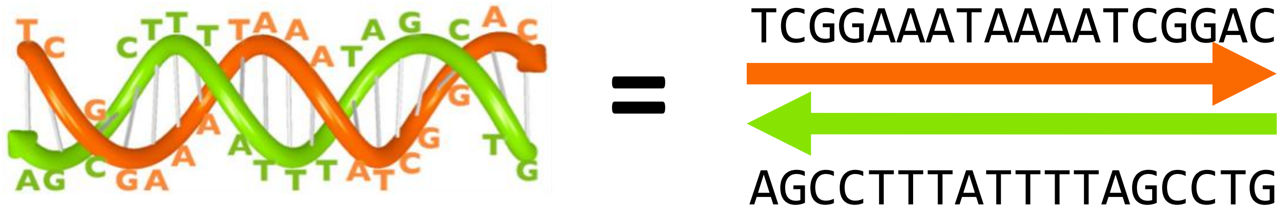
TCGGAAATAAAATCGGAC

AGCCTTTATTTTAGCCTG

DNA as a building material



U.S. National Library of Medicine

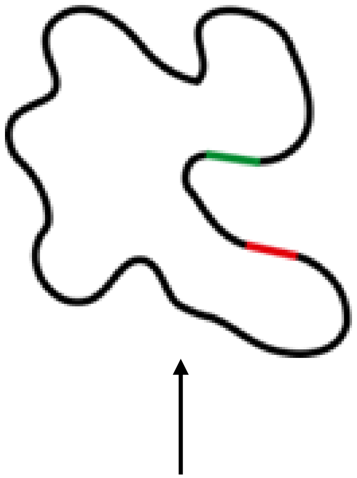


DNA strands can bind even if only part of strands are complementary:

DNA origami

Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

DNA origami

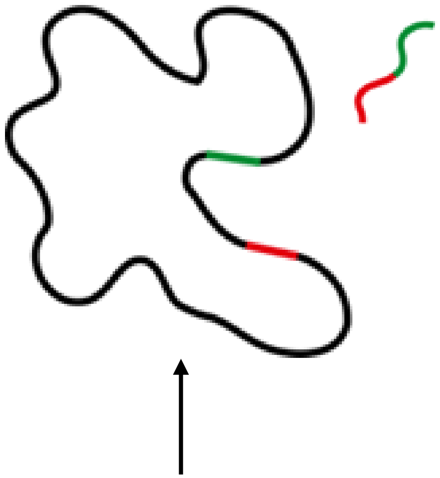


scaffold DNA strand

(M13mp18 bacteriophage virus)

Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
<u>Nature</u> 2006

DNA origami

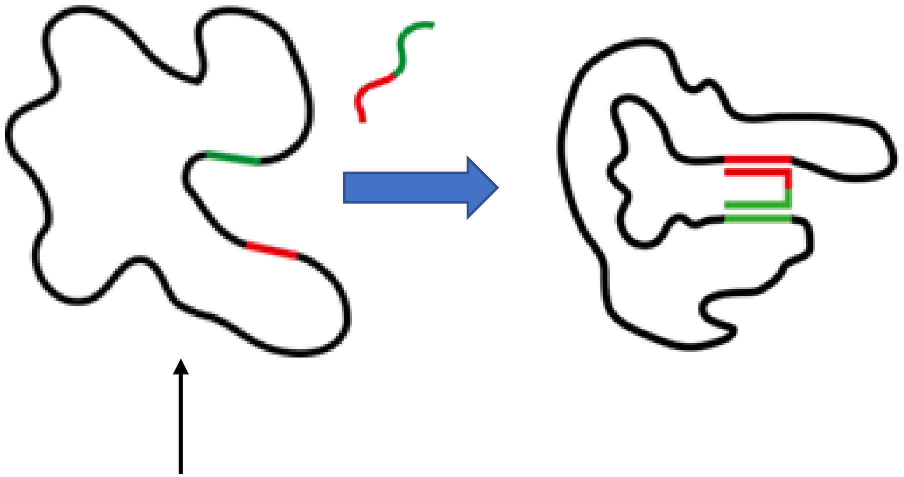


scaffold DNA strand

(M13mp18 bacteriophage virus)

Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

DNA origami

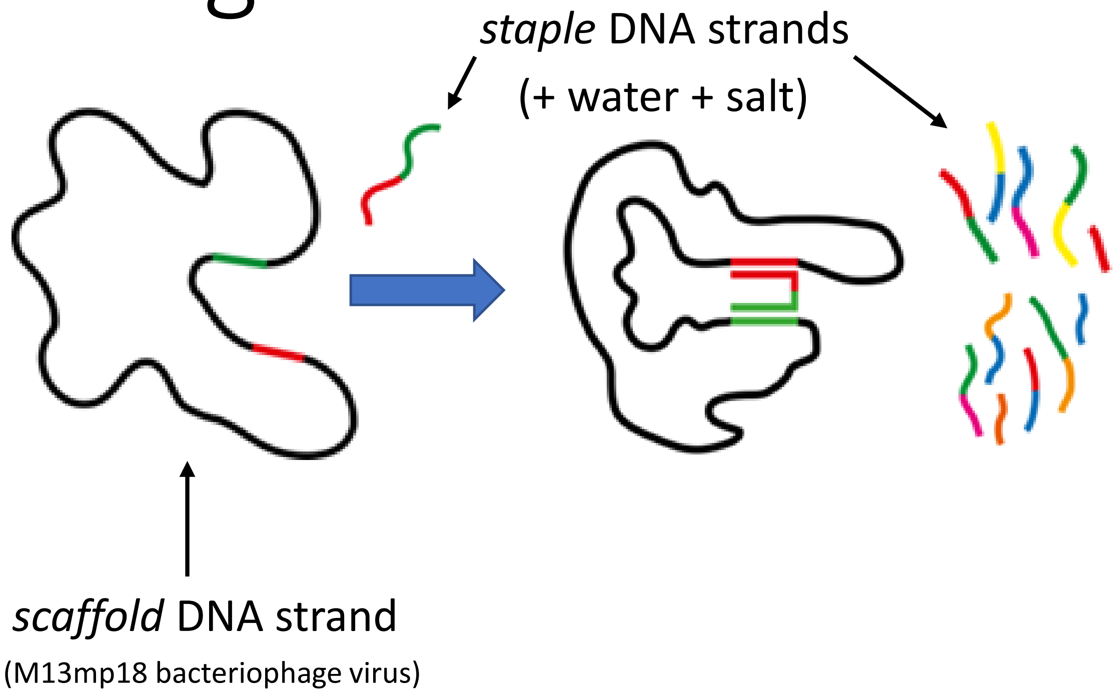


scaffold DNA strand

(M13mp18 bacteriophage virus)

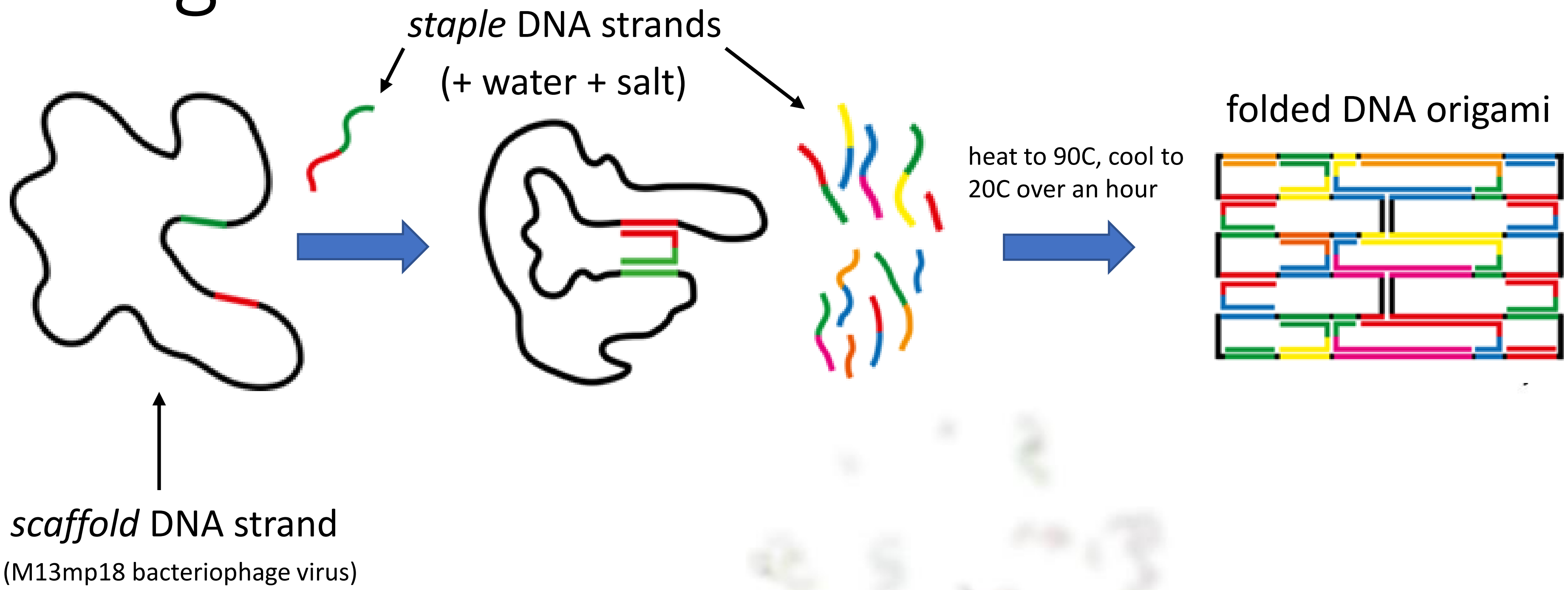
Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

DNA origami



Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

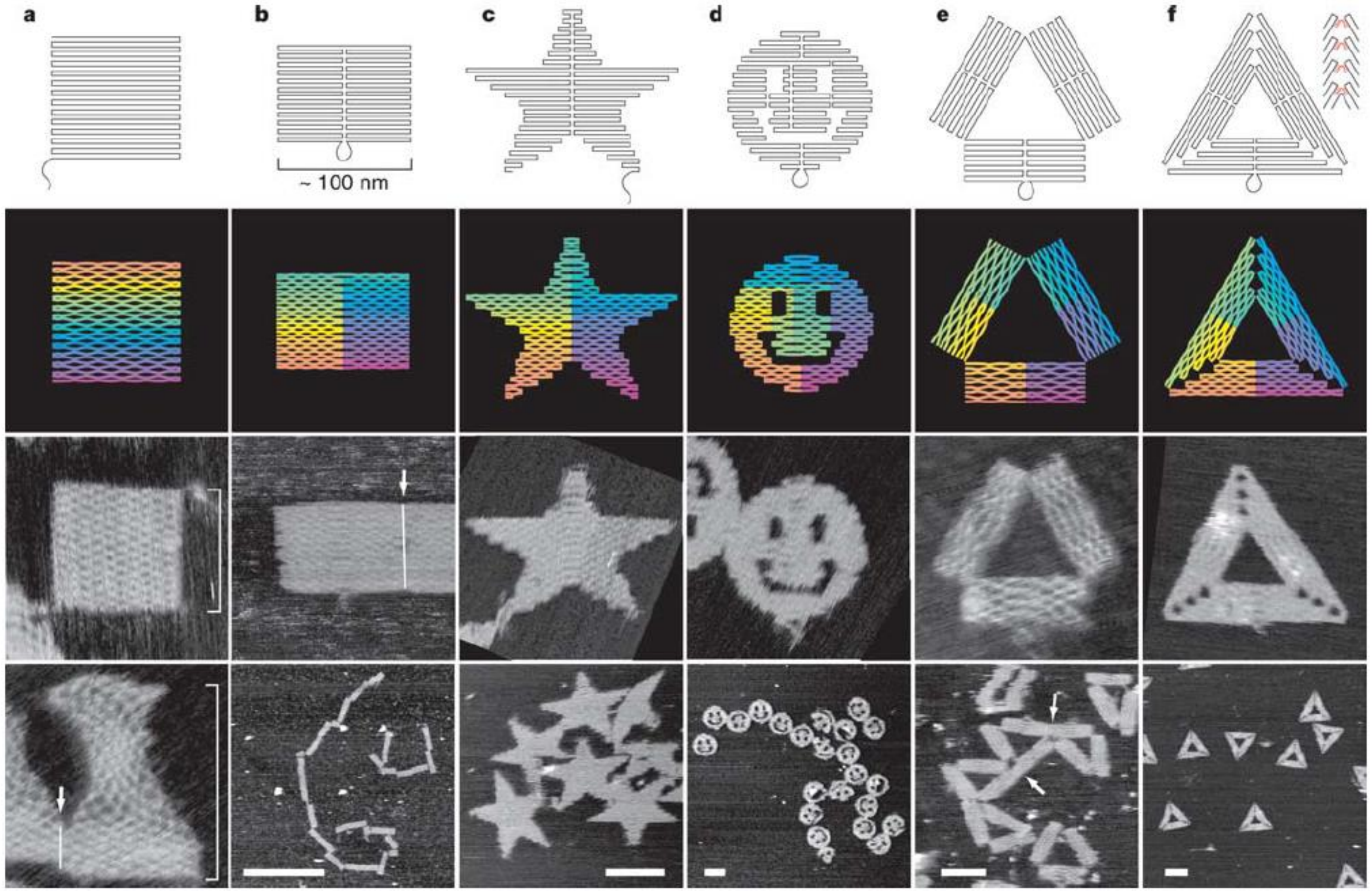
DNA origami



Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006

DNA origami

Paul Rothemund
Folding DNA to create nanoscale shapes and patterns
Nature 2006



Atomic force microscope images

100 nm

DNA nanotechnology applications

nonbiological:
- nanoscale resolution surface placement
- X-ray crystallization scaffolding
- molecular motors
- super-resolution imaging
- molecular circuits

biological:
- smart drugs
- mRNA detection
- cell surface marker detection
- genetically encoded structures

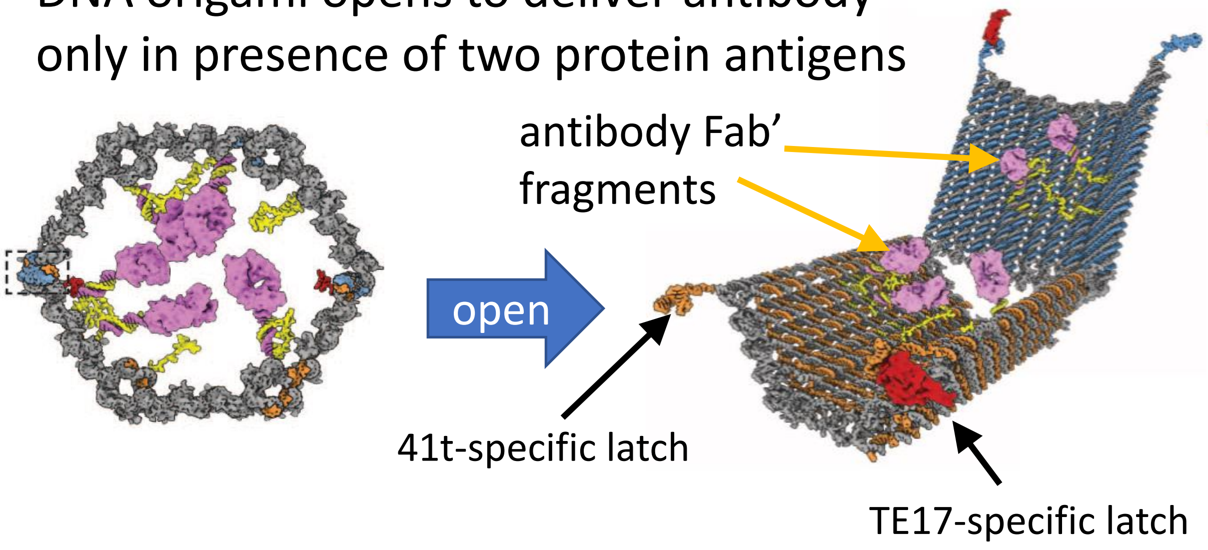
DNA nanotechnology applications

nonbiological:
- nanoscale resolution surface placement
- X-ray crystallization scaffolding
- molecular motors
- super-resolution imaging
- molecular circuits

example

biological:
- smart drugs
- mRNA detection
- cell surface marker detection
- genetically encoded structures

DNA origami opens to deliver antibody only in presence of two protein antigens

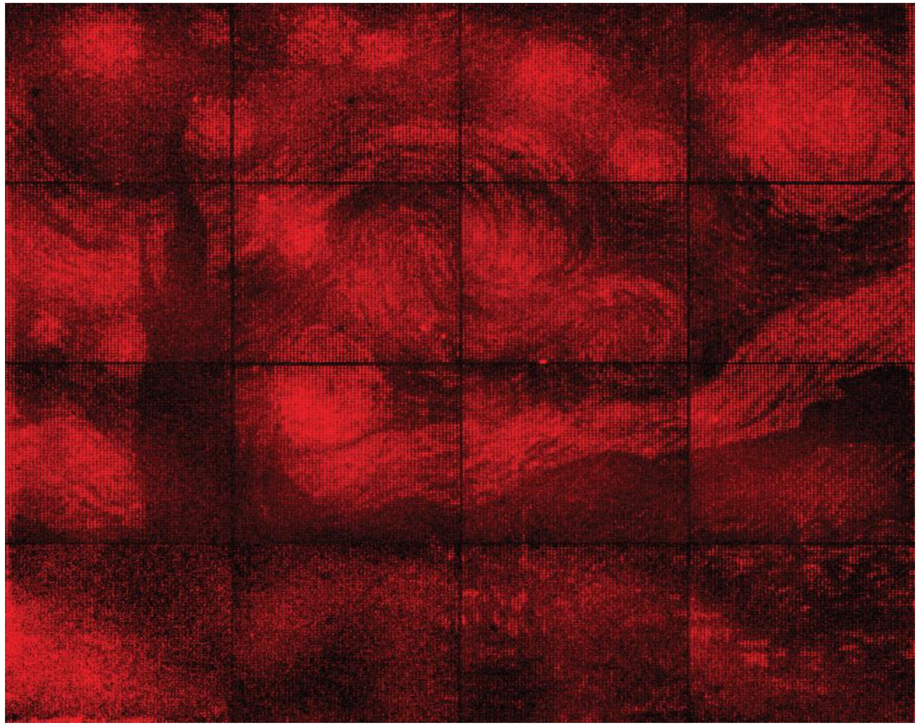


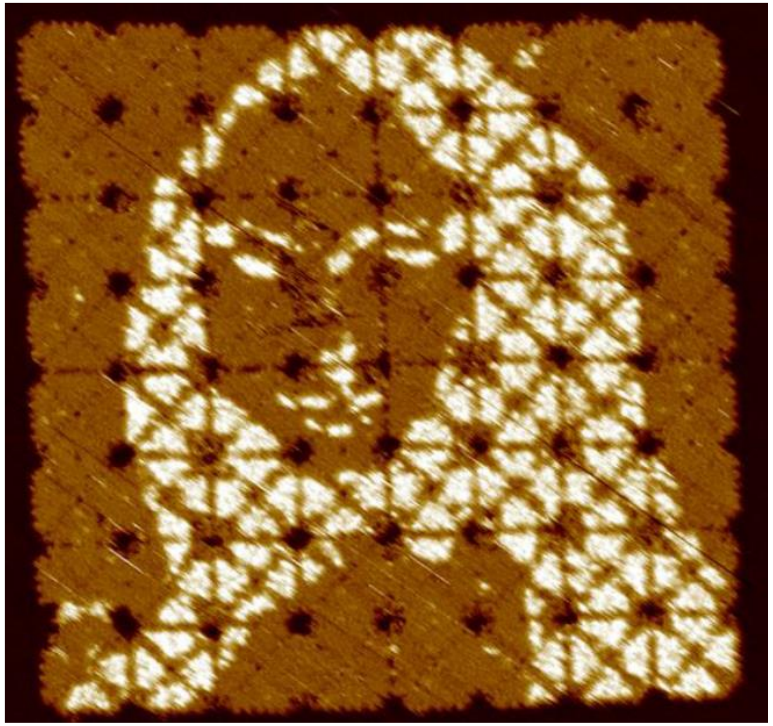
Shawn Douglas, Ido Bachelet, George Church, *A logic-gated nanorobot for targeted transport of molecular payloads*, Science 2012

DNA nanotechnology applications

nonbiological:

- **art**



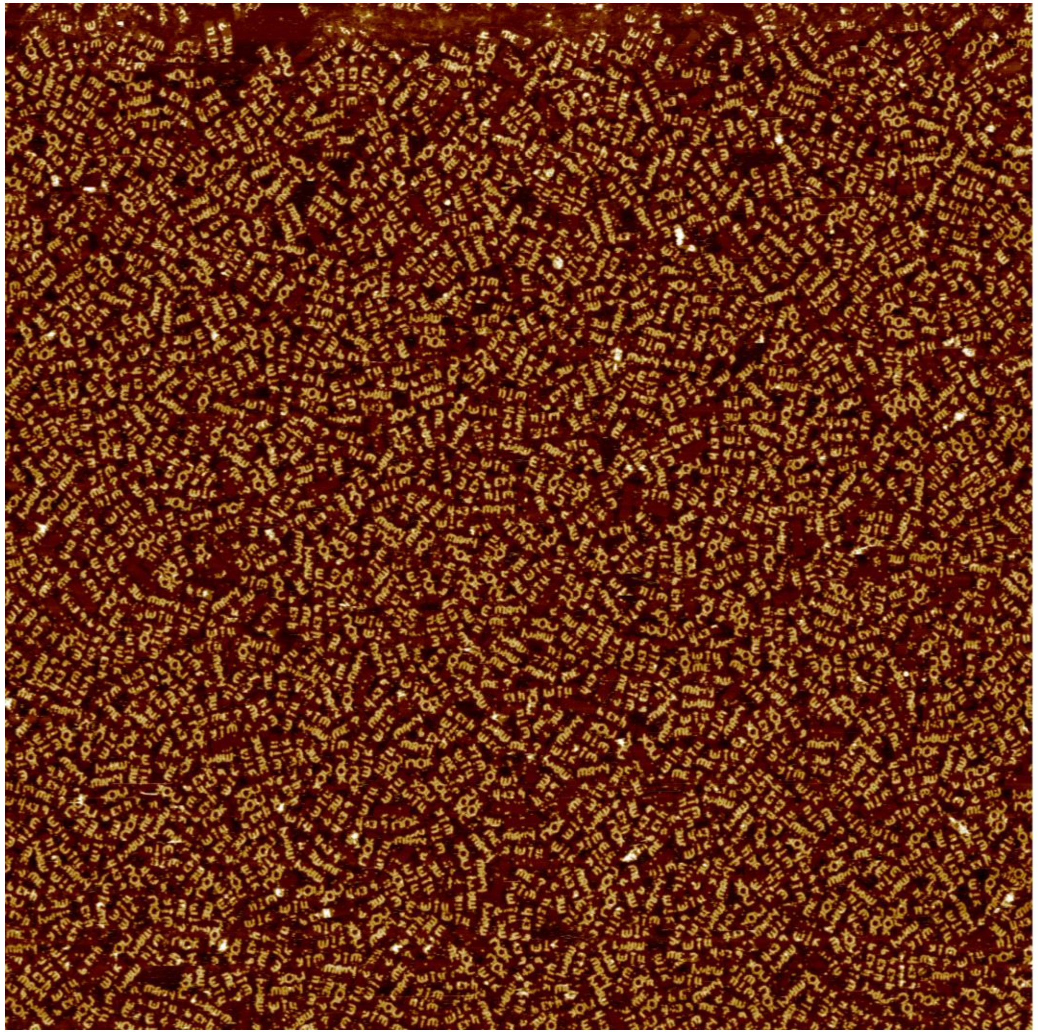


Ashwin Gopinath, Evan Miyazono, Andrei Faraon, Paul Rothemund, *Engineering and mapping nanocavity emission via precision placement of DNA origami*, Nature 2016

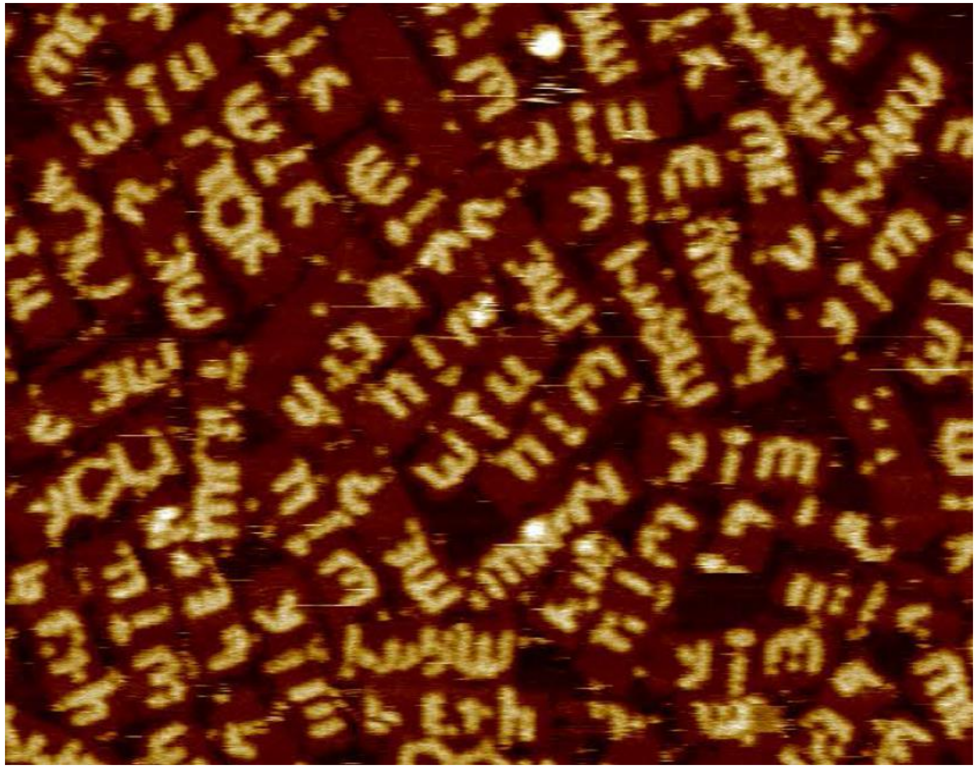
Grigory Tikhomirov, Philip Petersen, and Lulu Qian. *Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns*. Nature 2017.

Other applications of DNA nanotechnology

4 μm wide scan



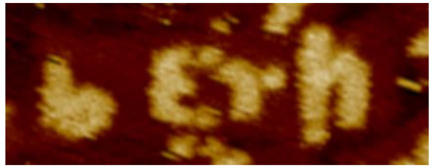
zoom in

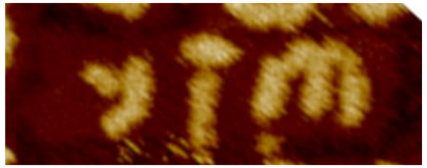


Cherry-picked samples

100 nm

Beth Yim





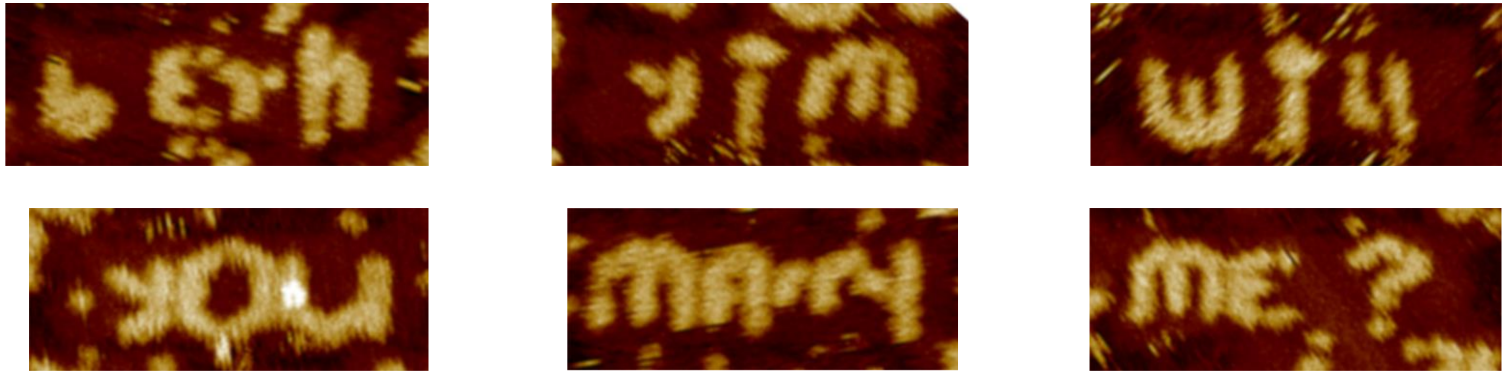
Cherry-picked samples

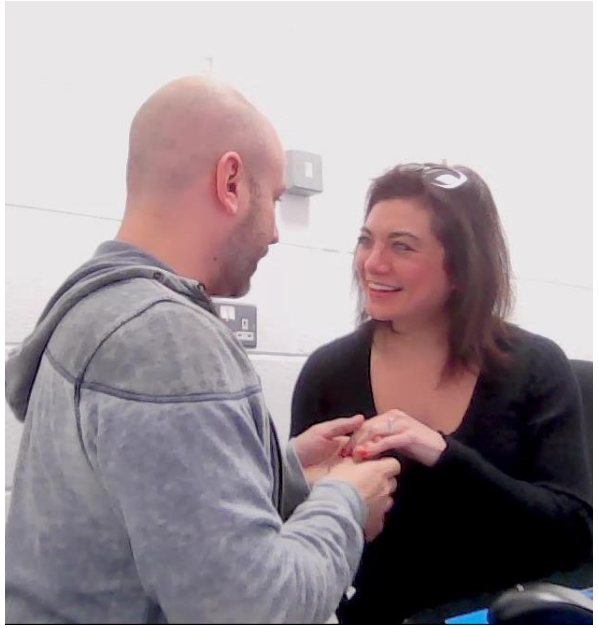
Beth Yim

Cherry-picked samples

100 nm

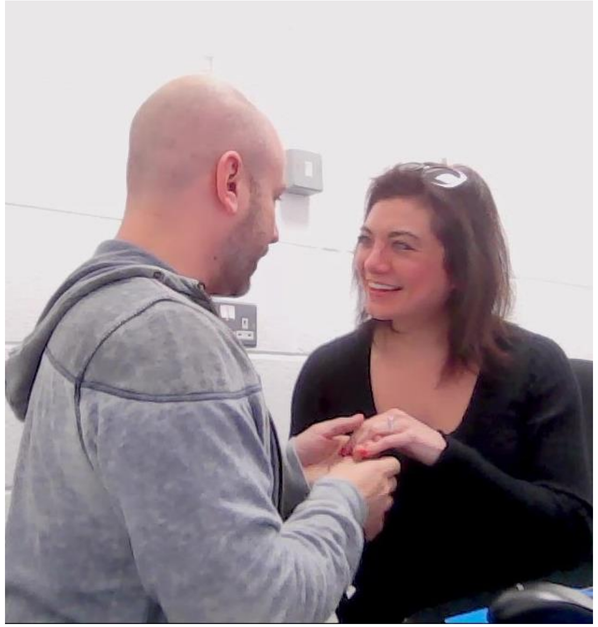
Beth Yim





Cherry-picked samples

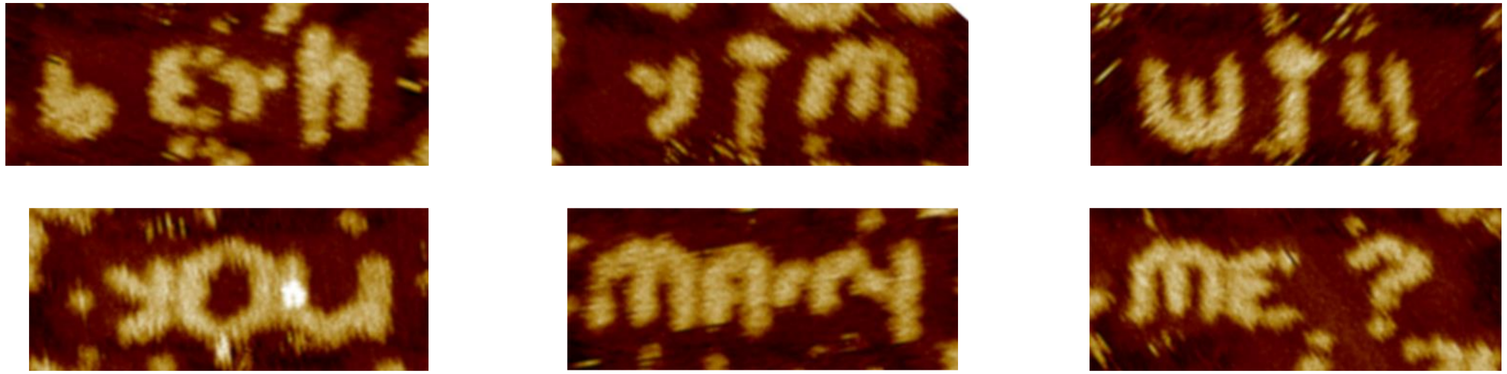
Beth Yim

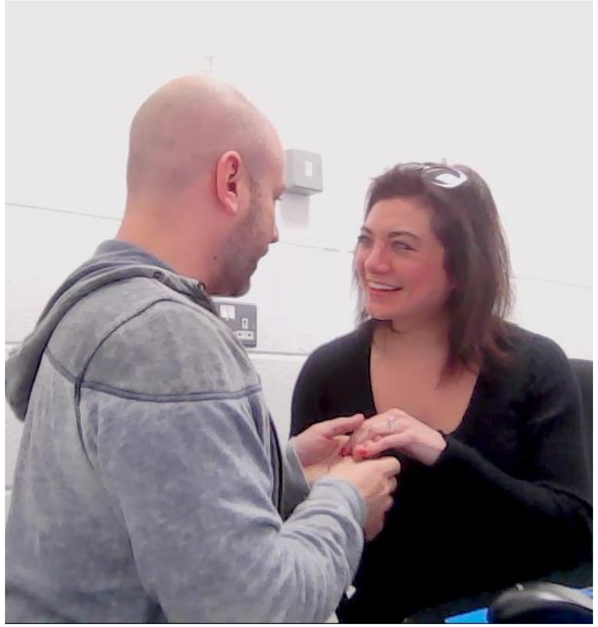


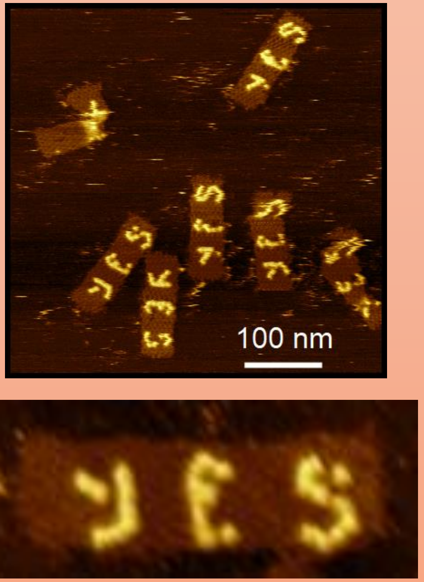
Cherry-picked samples

100 nm

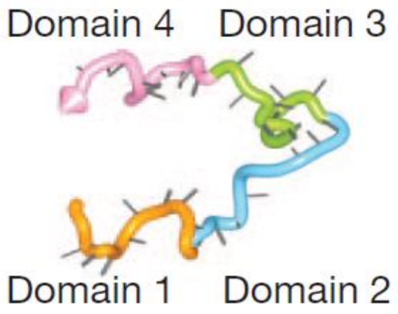
Beth Yim

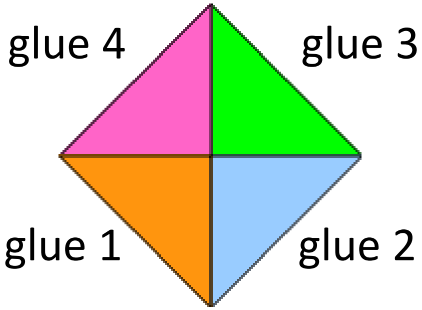




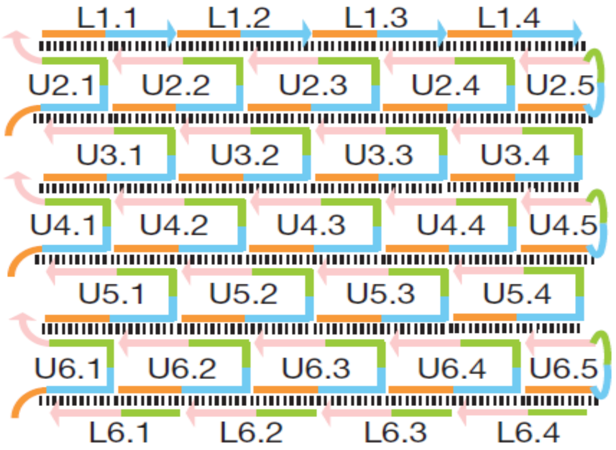


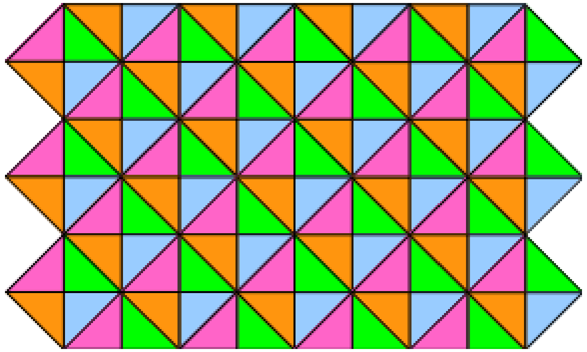
DNA single-stranded tiles





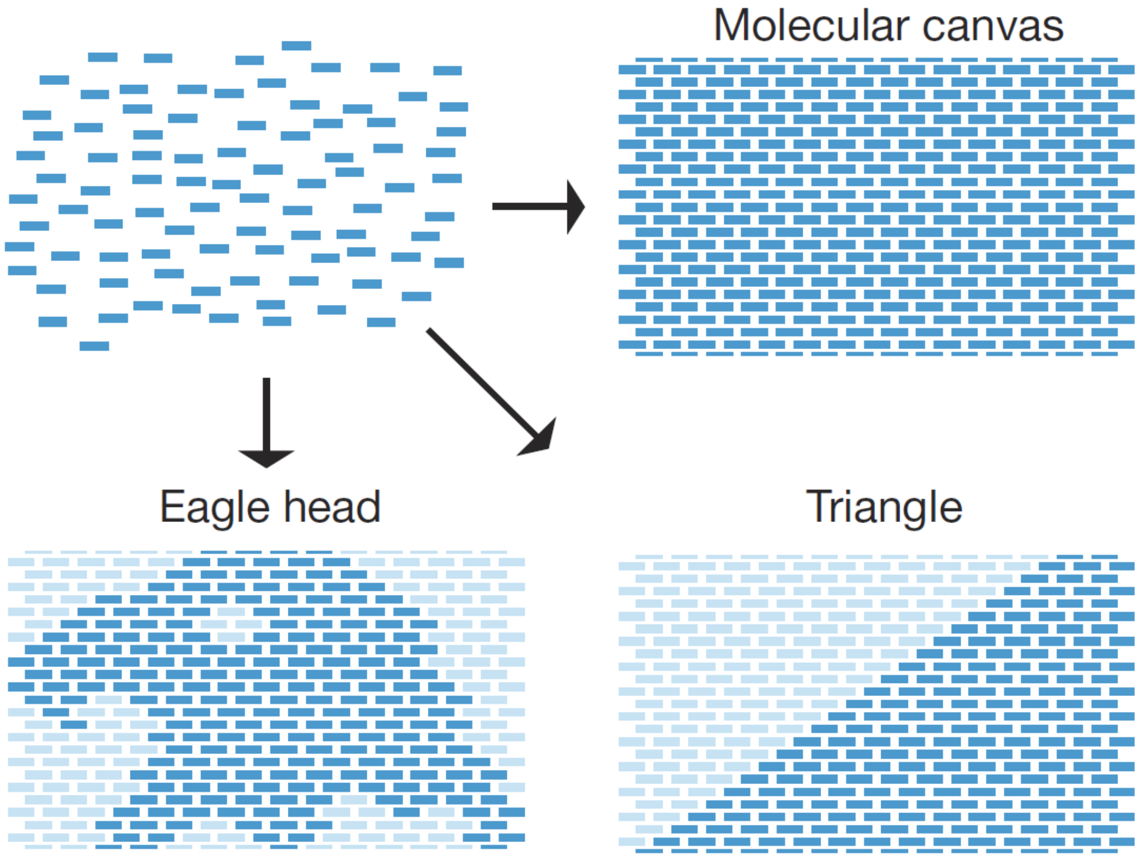
assembly





Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif
Programming DNA tube circumferences
Science 321, 824-826 (2008)

Single-stranded tiles for making any shape



Bryan Wei, Mingjie Dai, and Peng Yin. *Complex shapes self-assembled from single-stranded DNA tiles*. Nature 2012.

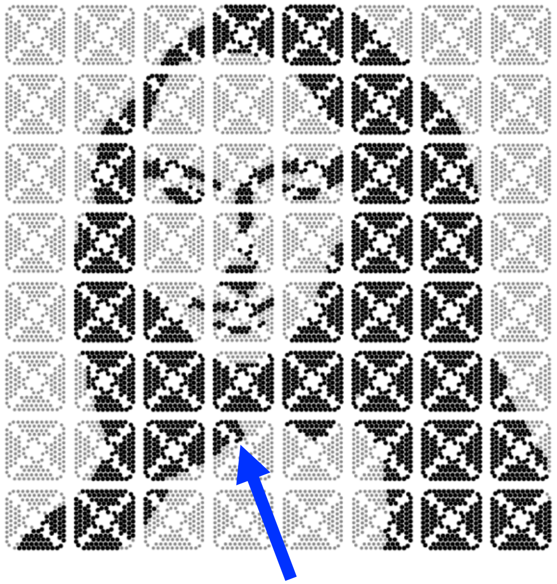
Uniquely addressed self-assembly versus algorithmic

Unique addressing: each DNA "monomer" appears **exactly once** in final structure:

single DNA origami

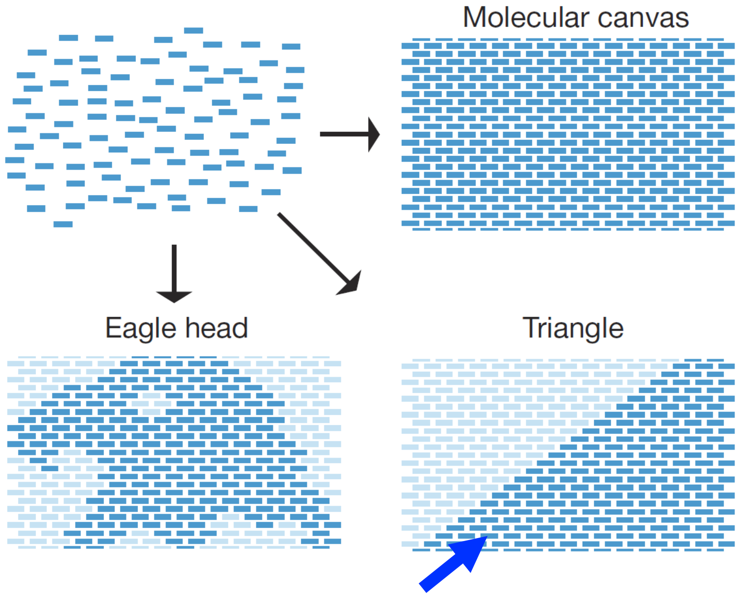
staple strand for position (4,2)

array of many DNA origamis



origami for position (4,2)

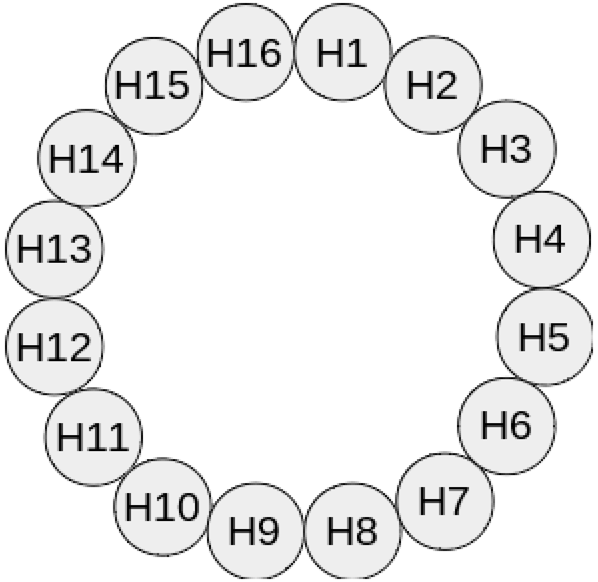
uniquely-addressed tiles



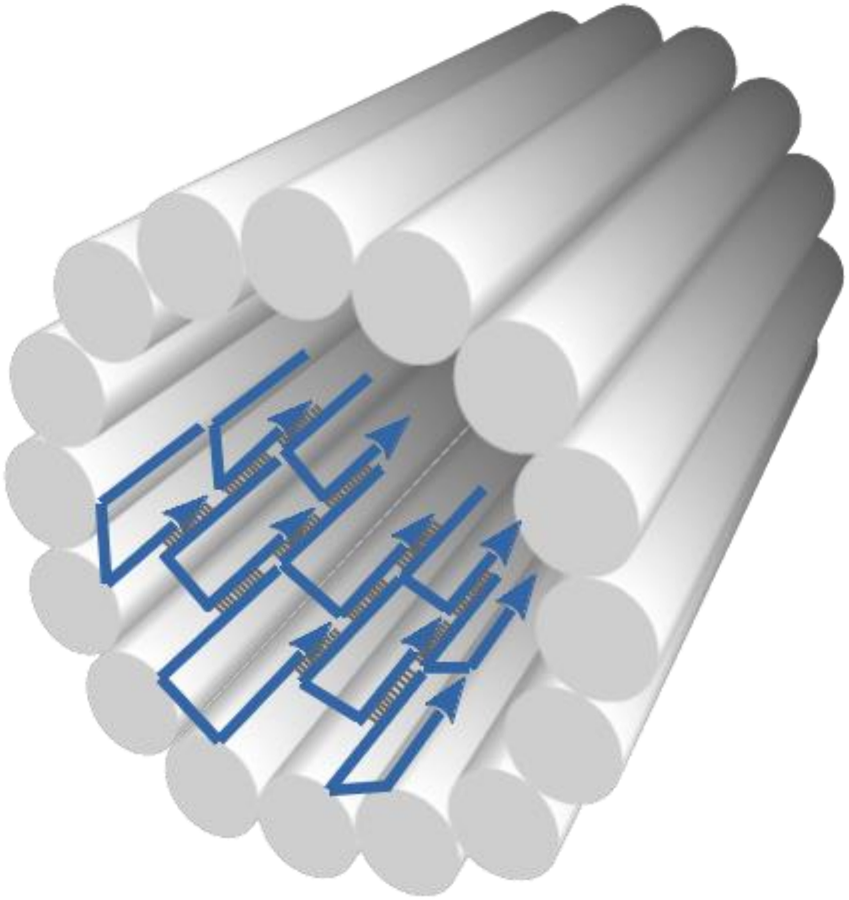
tile for position (4,2)

Algorithmic: DNA tiles are **reused** throughout the structure.

Single-stranded tile tubes



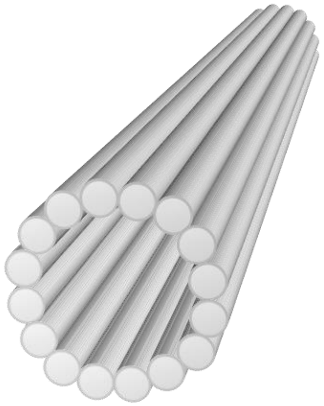
DNA-level diagram of 20-helix tube



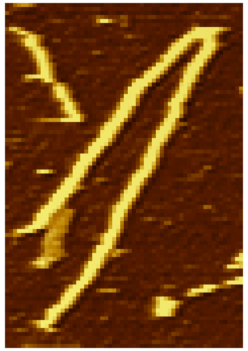
Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif, Programming DNA tube circumferences, *Science* 321, 824-826 (2008)

Tubes and ribbons

tube



AFM
image

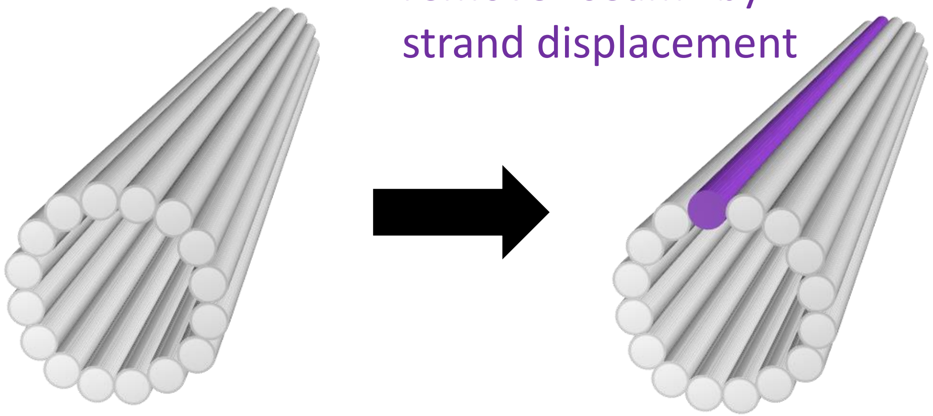


500 nm

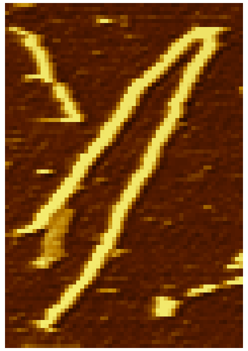
Tubes and ribbons

tube

remove “seam” by
strand displacement



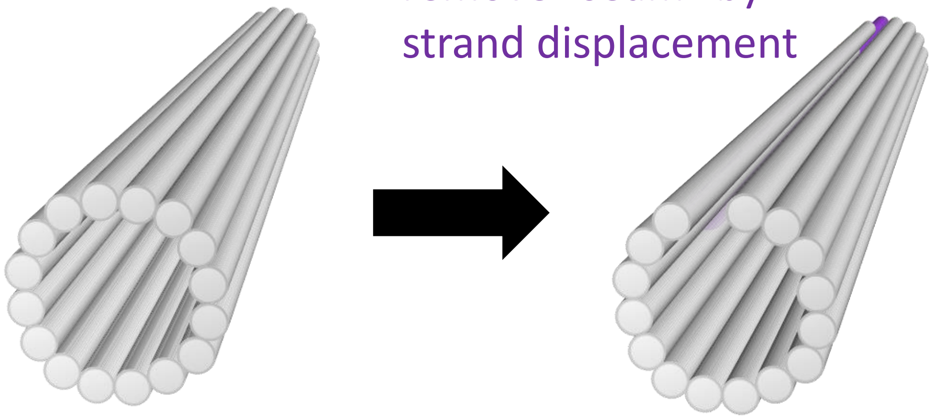
AFM
image



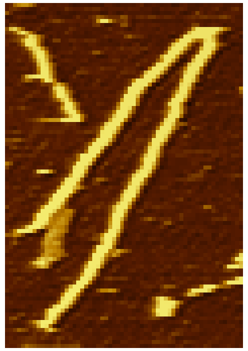
500 nm

Tubes and ribbons

tube

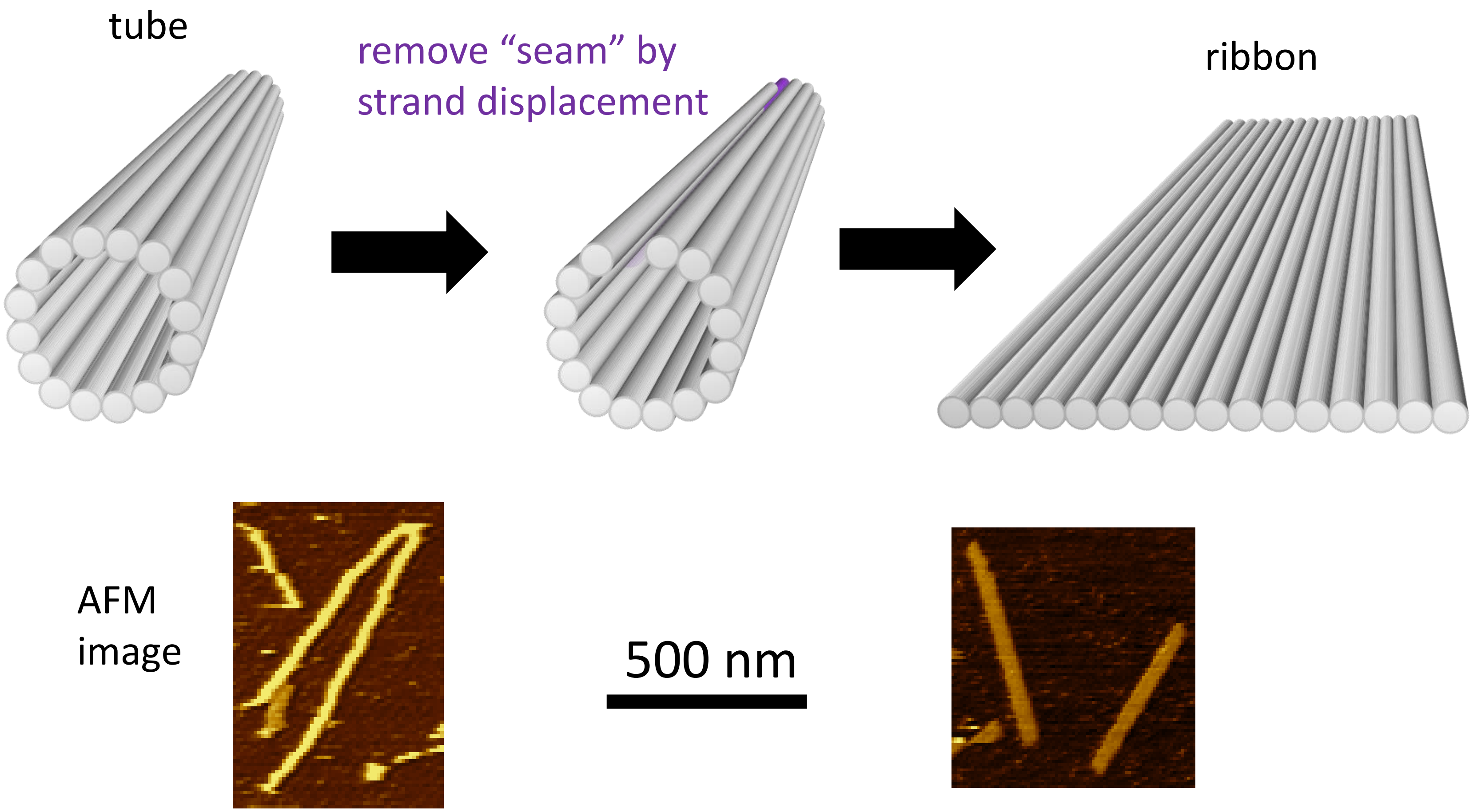


AFM
image



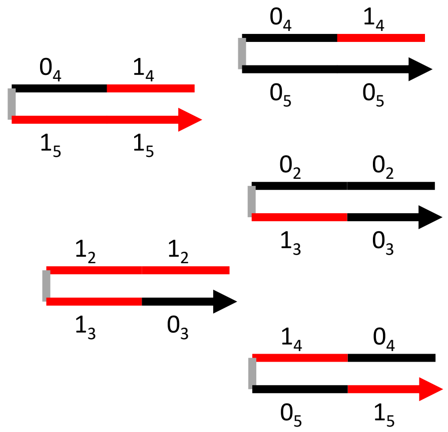
500 nm

Tubes and ribbons



Seeded growth

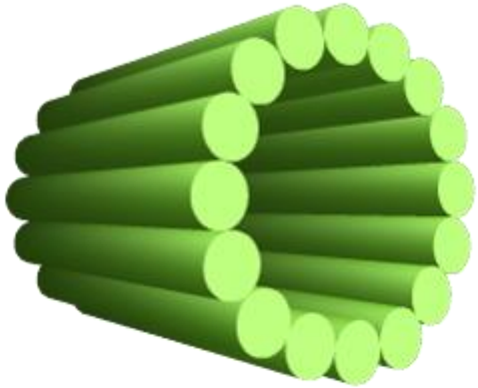
single-stranded tiles
implementing circuit gates



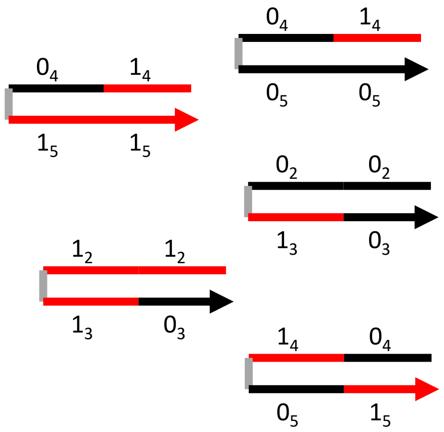
barrier to *nucleation* (tile growth without seed) at [tile]=100 nM and temperature=50.9° C

Seeded growth

DNA origami seed



single-stranded tiles
implementing circuit gates

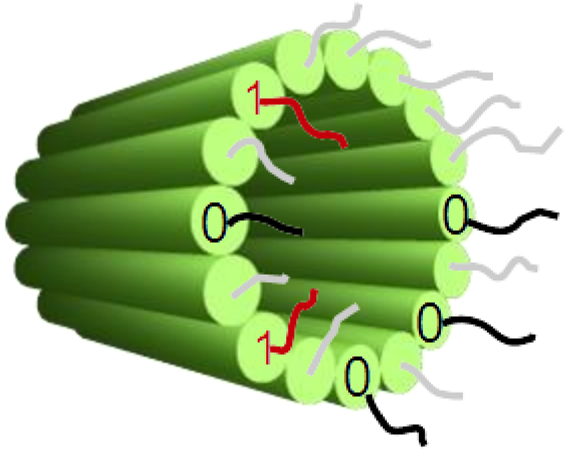


barrier to *nucleation* (tile
growth without seed) at
[tile]=100 nM and
temperature=50.9° C

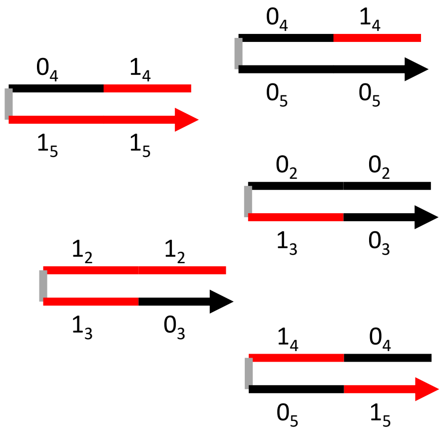
Seeded growth

DNA origami seed

single-stranded “input-adapter” extensions encoding 6 input bits



single-stranded tiles implementing circuit gates

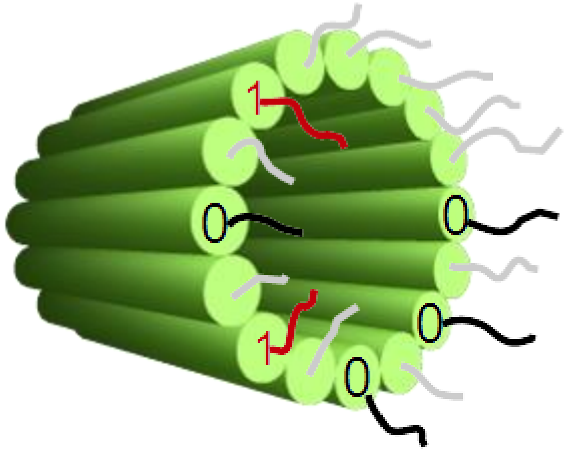


barrier to *nucleation* (tile growth without seed) at [tile]=100 nM and temperature=50.9° C

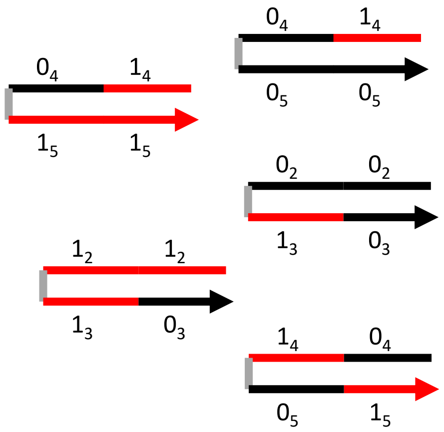
Seeded growth

DNA origami seed

single-stranded "input-adapter" extensions encoding 6 input bits

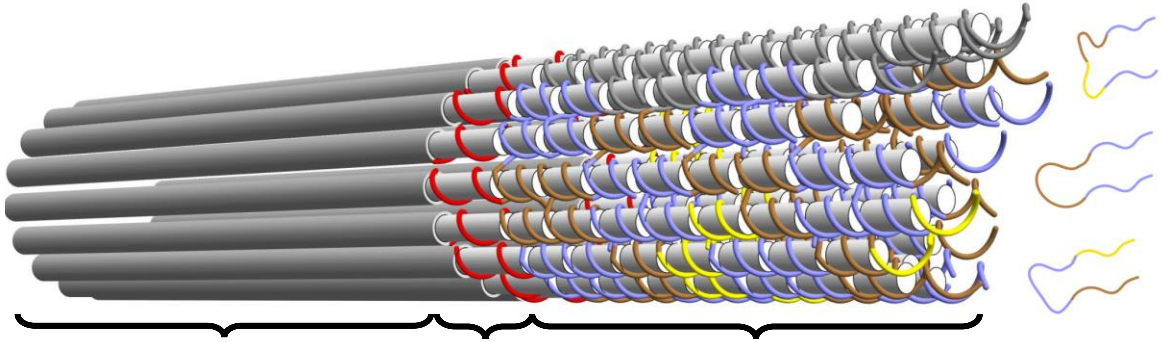


single-stranded tiles implementing circuit gates



barrier to *nucleation* (tile growth without seed) at [tile]=100 nM and temperature=50.9° C

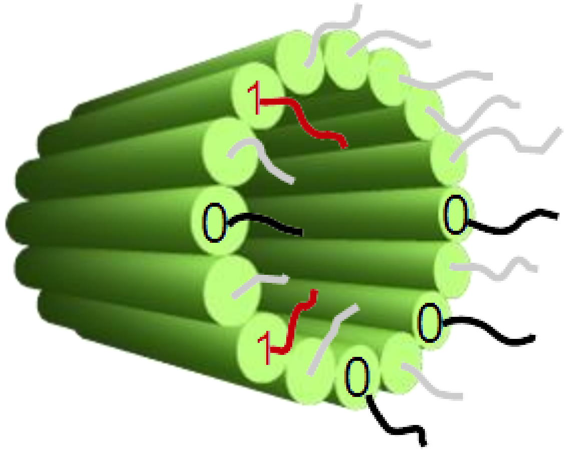
hold 8-48 hours



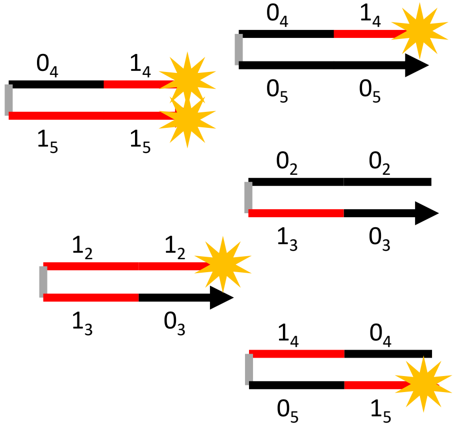
Seeded growth

DNA origami seed

single-stranded “input-adapter” extensions encoding 6 input bits



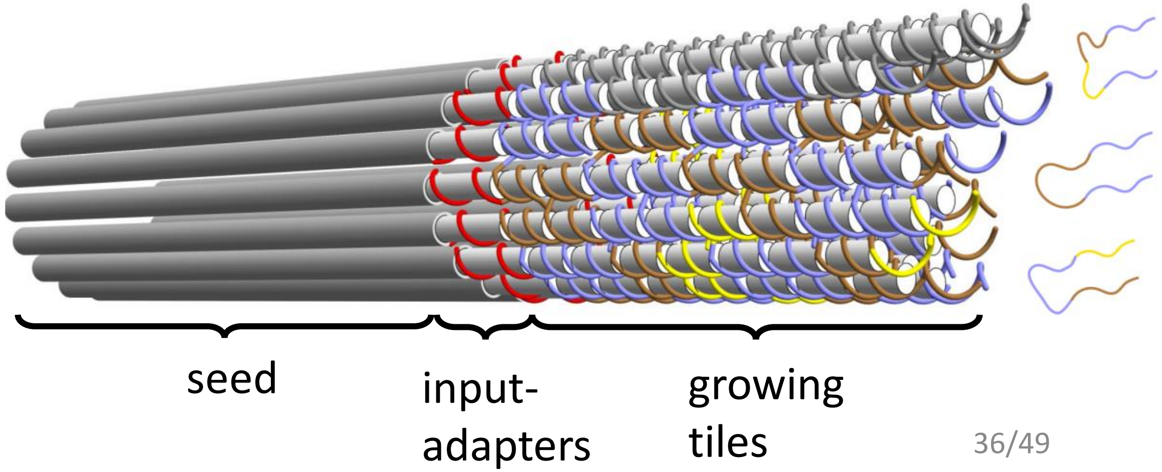
single-stranded tiles implementing circuit gates



biotins if output = 1

barrier to *nucleation* (tile growth without seed) at [tile]=100 nM and temperature=50.9° C

hold 8-48 hours

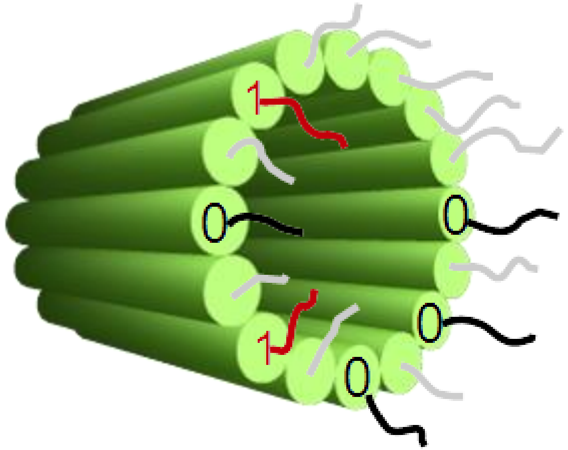


Seeded growth

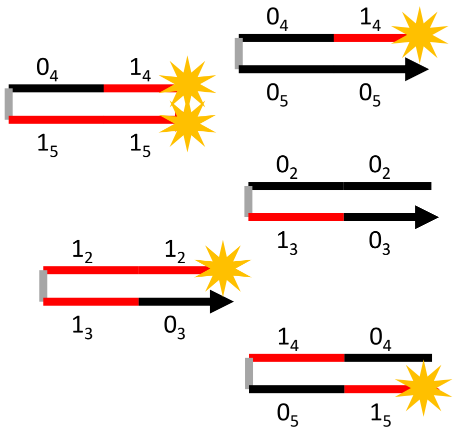
can later add streptavidin to bind biotins and visualize where the 1's are

DNA origami seed

single-stranded "input-adapter" extensions encoding 6 input bits



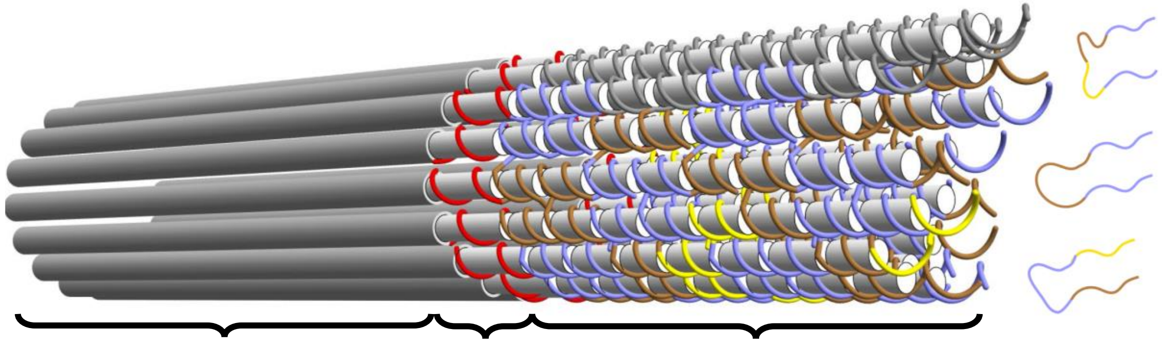
single-stranded tiles implementing circuit gates



biotins if output = 1

barrier to *nucleation* (tile growth without seed) at [tile]=100 nM and temperature=50.9° C

hold 8-48 hours



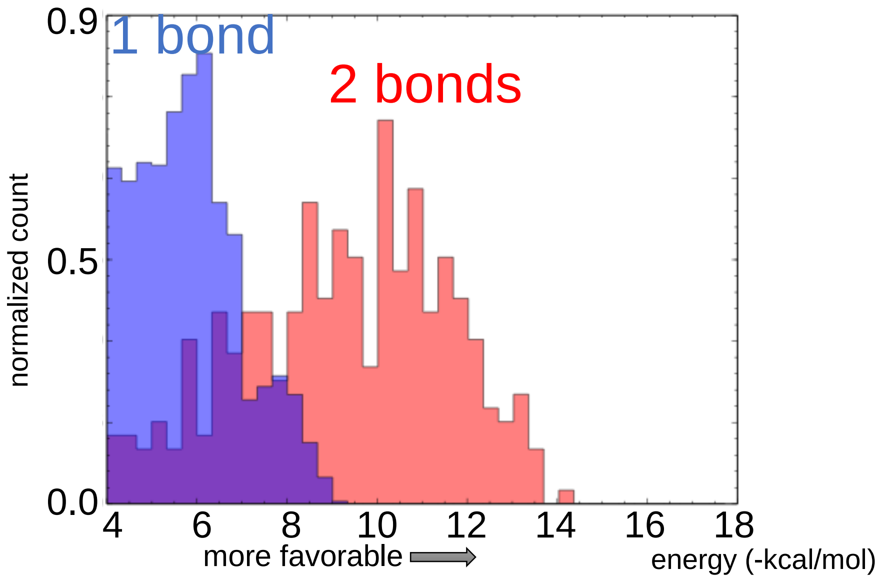
seed

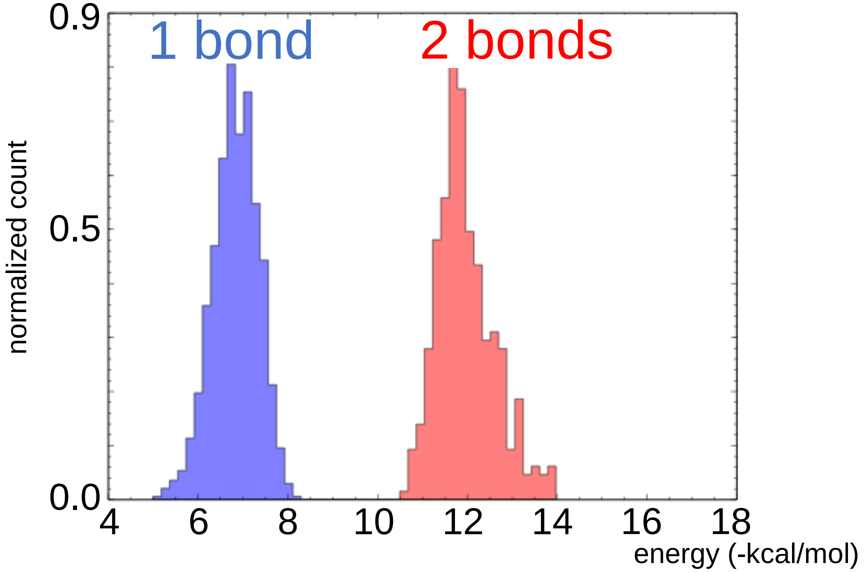
input-adapters

growing tiles

DNA sequence design

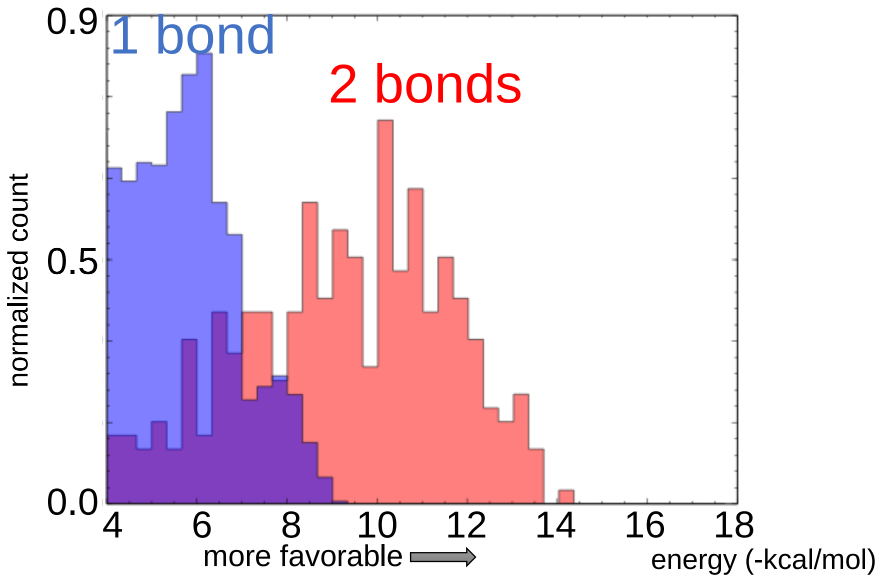
Random sequences vs designed sequences

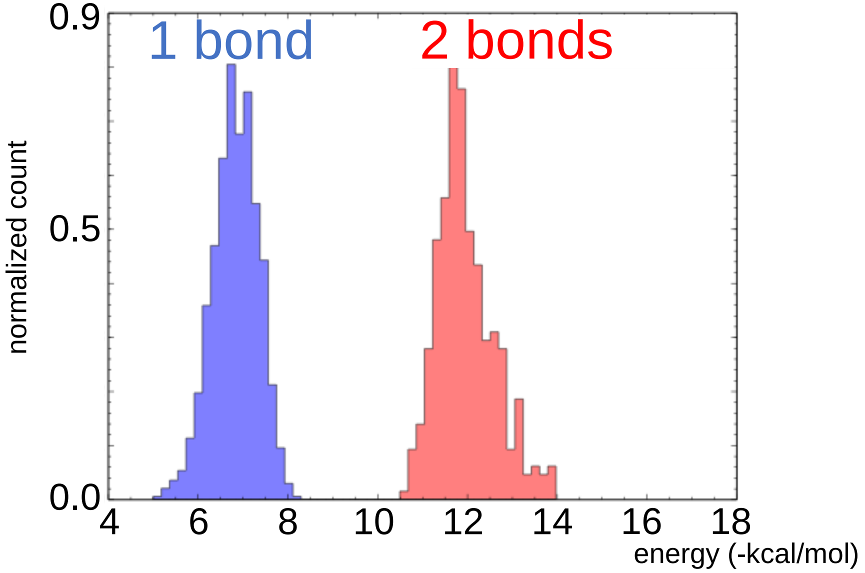


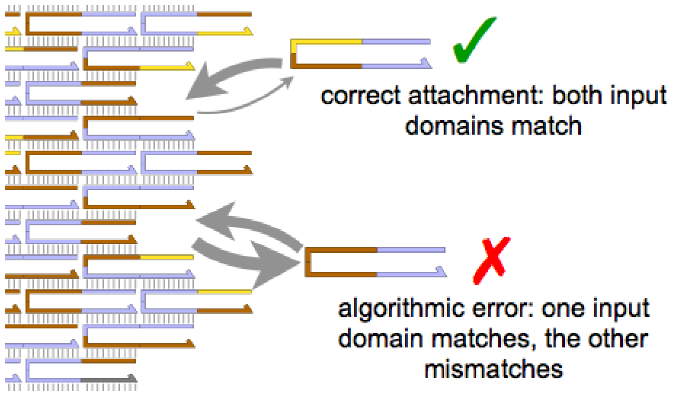


DNA sequence design

Random sequences vs designed sequences

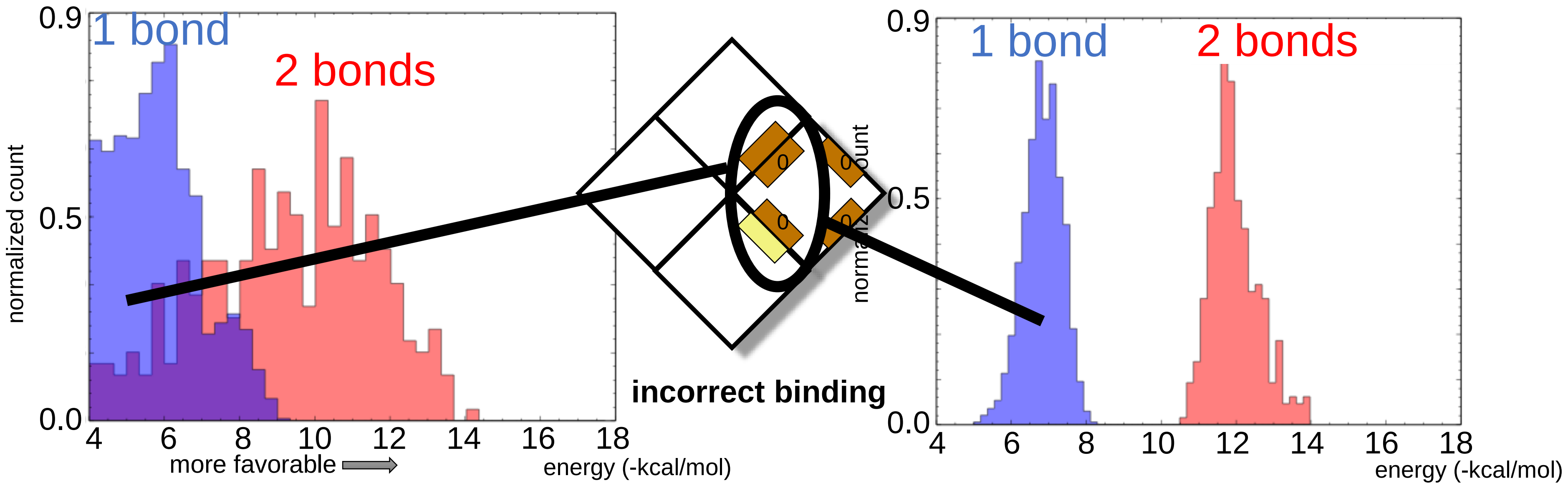


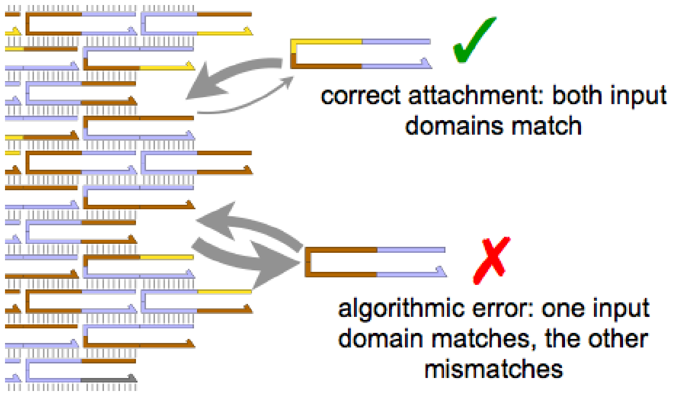




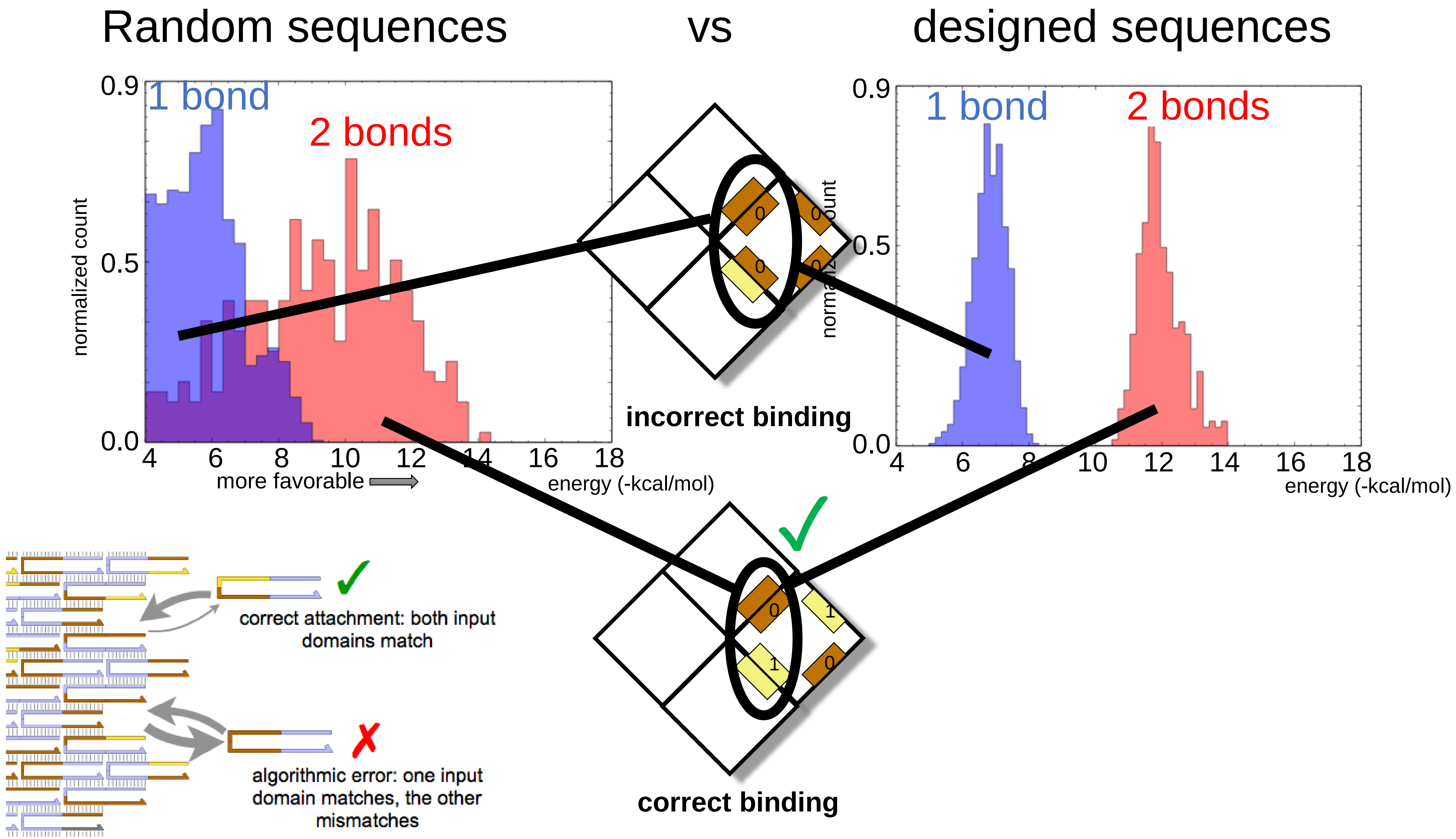
DNA sequence design

Random sequences vs designed sequences



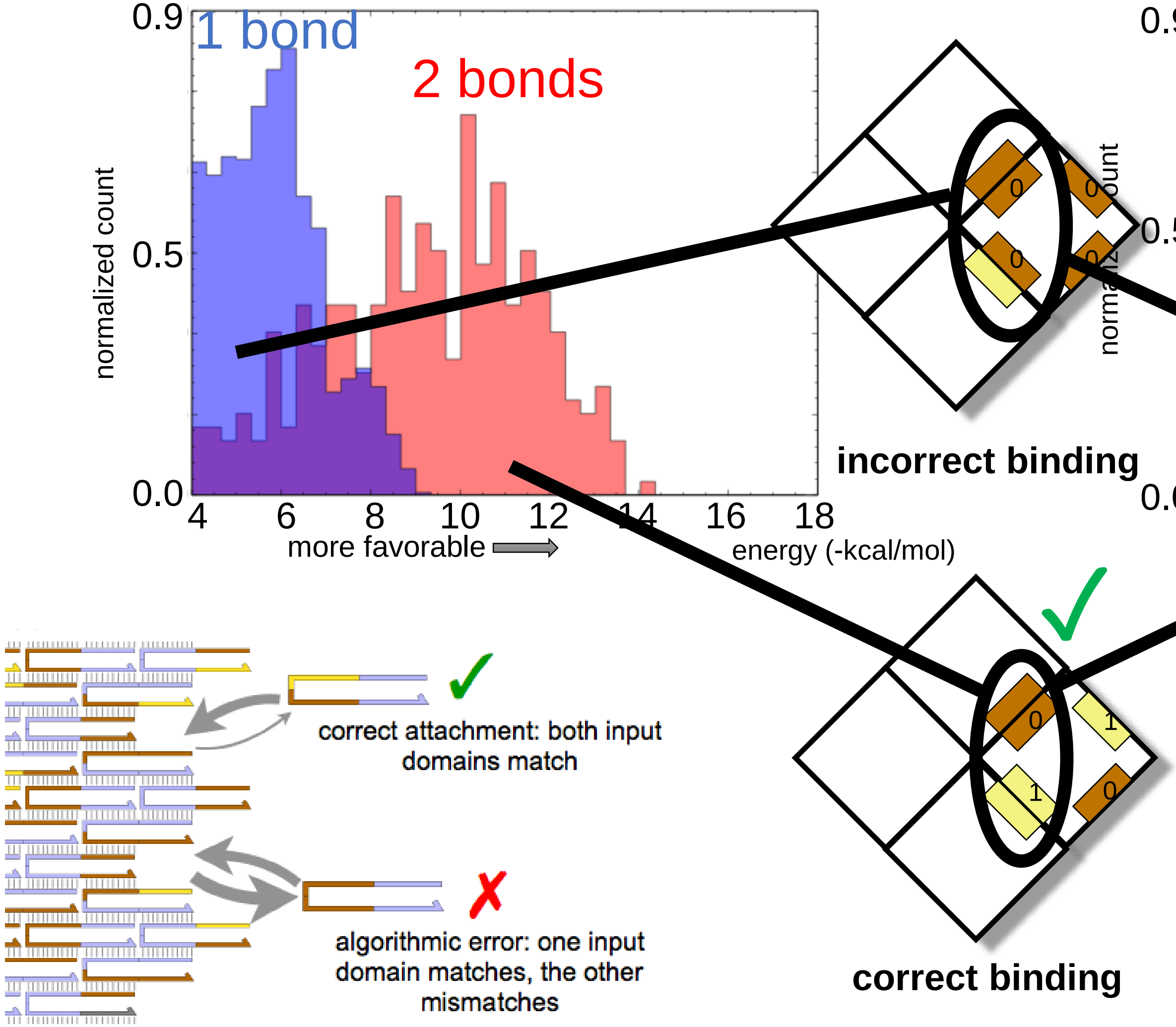


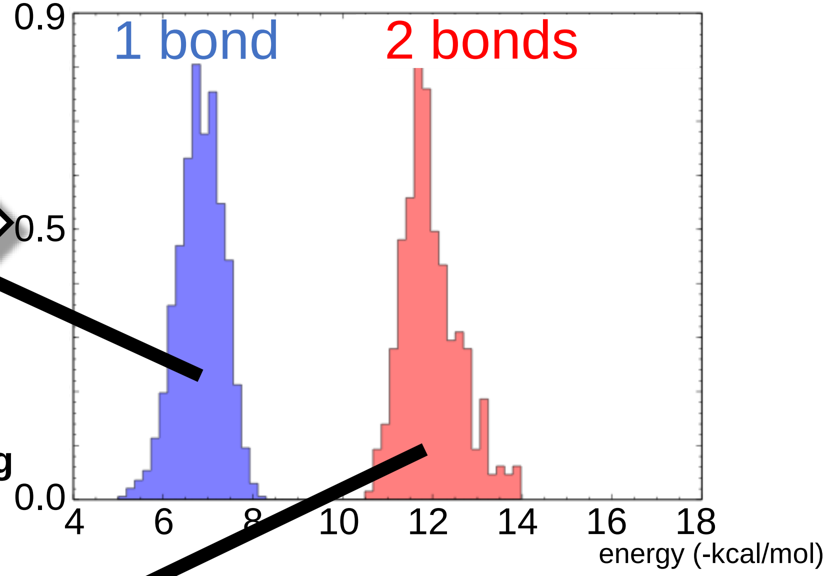
DNA sequence design

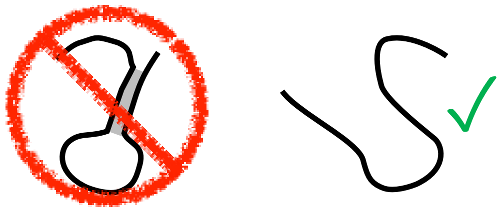


DNA sequence design

Random sequences vs designed sequences



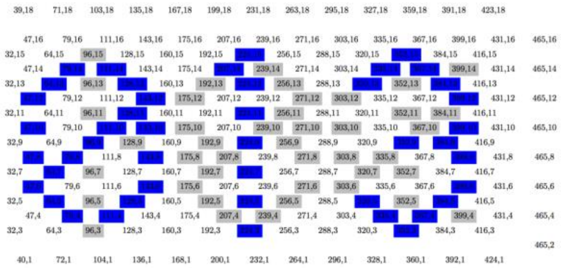




Other goals:

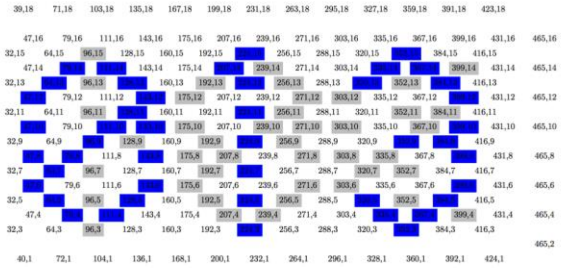
- low strand secondary structure
- low interaction between strands

Bar-coding origami seed for imaging multiple samples at once



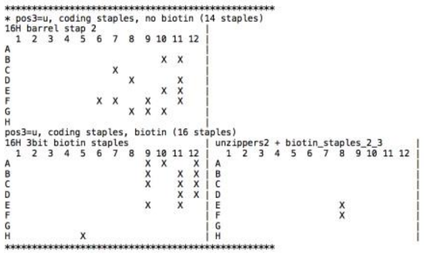
some staples of origami seed have version with a biotin

Bar-coding origami seed for imaging multiple samples at once



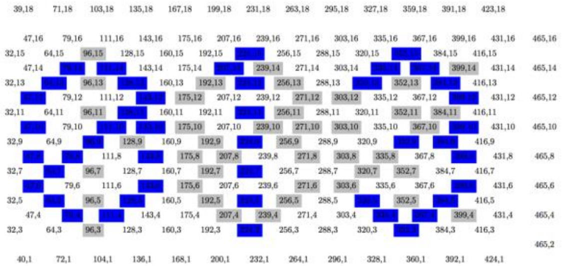
some staples of origami seed have version with a biotin

Generate plate map



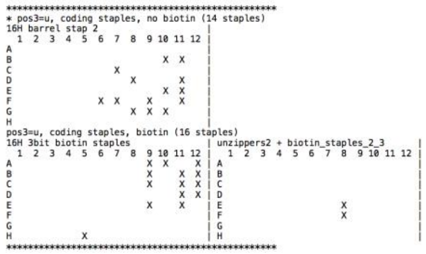
```
**************************************************
* pos3=u, coding staples, no biotin (14 staples)
16H barrel stap 2                          |
  1  2  3  4  5  6  7  8  9 10 11 12       |
A                                          |
B                          X  X            |
C                    X                     |
D                       X        X         |
E                          X  X            |
F                 X  X     X     X         |
G                       X  X  X            |
H                                          |
pos3=u, coding staples, biotin (16 staples)
16H 3bit biotin staples                    | unzippers2 + biotin_staples_2_3        |
  1  2  3  4  5  6  7  8  9 10 11 12       |   1  2  3  4  5  6  7  8  9 10 11 12  |
A                          X  X     X      | A                                      |
B                          X     X  X      | B                                      |
C                          X     X  X      | C                                      |
D                                X  X      | D                                      |
E                          X     X         | E                      X               |
F                                          | F                      X               |
G                                          | G                                      |
H              X                           | H                                      |
**************************************************
```

Bar-coding origami seed for imaging multiple samples at once



some staples of origami seed
have version with a biotin

Generate
plate map

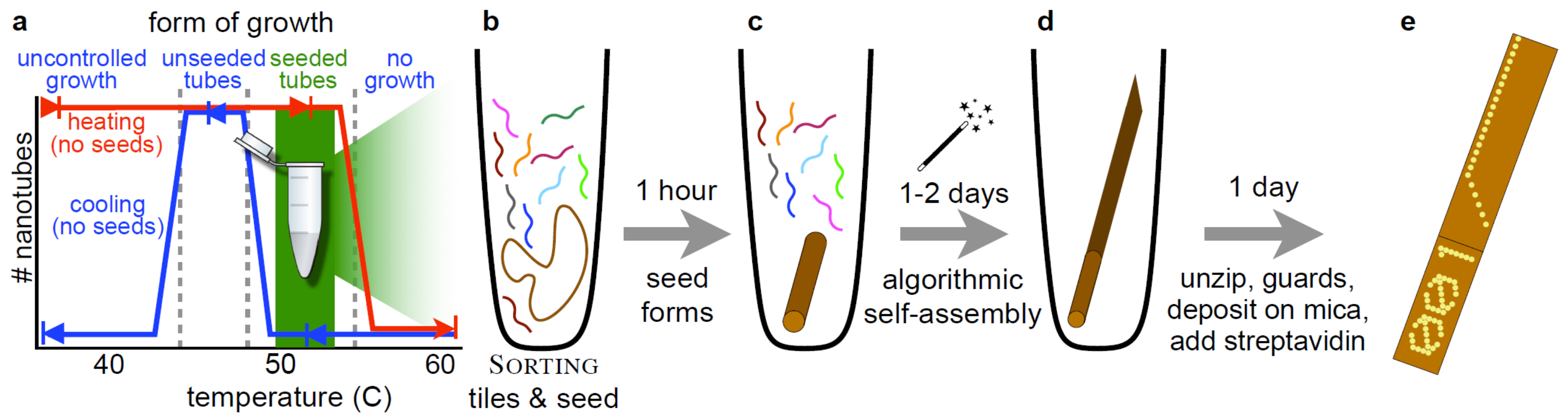


```
***************************************************
* pos3=u, coding staples, no biotin (14 staples)
16H barrel stap 2                                |
  1  2  3  4  5  6  7  8  9 10 11 12 |
A                                    |
B                          X  X      |
C                    X               |
D                       X        X   |
E                          X  X      |
F                 X  X     X     X   |
G                       X  X  X      |
H                                    |
pos3=u, coding staples, biotin (16 staples)
16H 3bit biotin staples              | unzippers2 + biotin_staples_2_3 |
  1  2  3  4  5  6  7  8  9 10 11 12 |   1  2  3  4  5  6  7  8  9 10 11 12 |
A                          X  X     X | A                                    |
B                          X     X  X | B                                    |
C                          X     X  X | C                                    |
D                                X  X | D                                    |
E                          X     X    | E                     X              |
F                                     | F                     X              |
G                                     | G                                    |
H              X                      | H                                    |
***************************************************
```

represents some combination of
circuit and input, e.g.,
013 = *"parity circuit, input=011010"*

label with
streptavidin

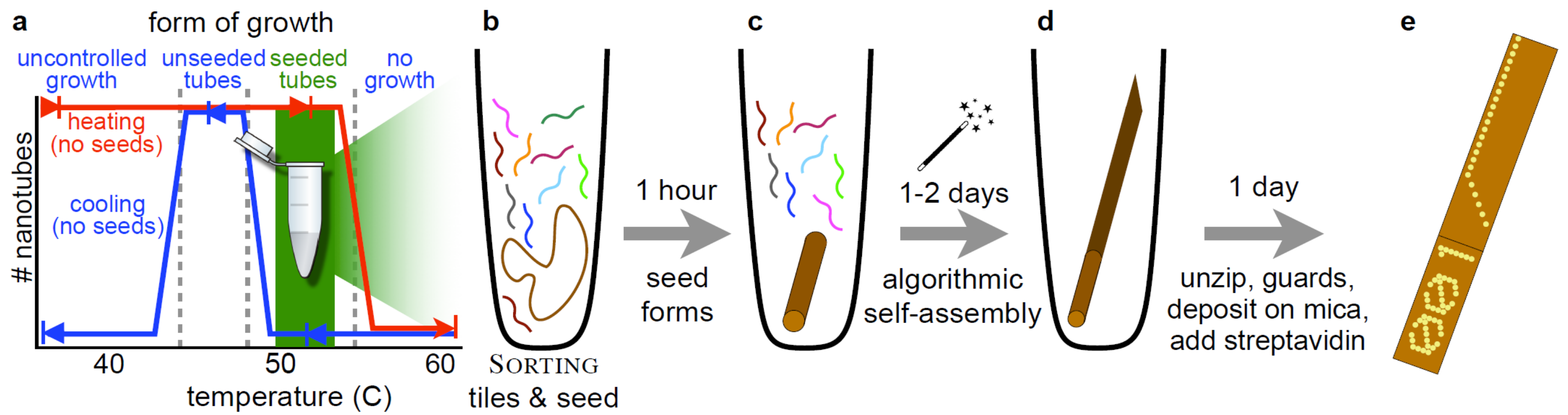
Experimental protocol



To execute circuit γ on input $x \in \{0,1\}^*$:

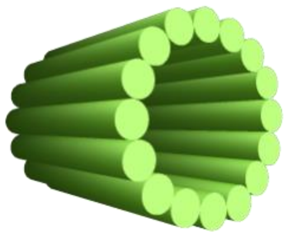
- Mix

Experimental protocol

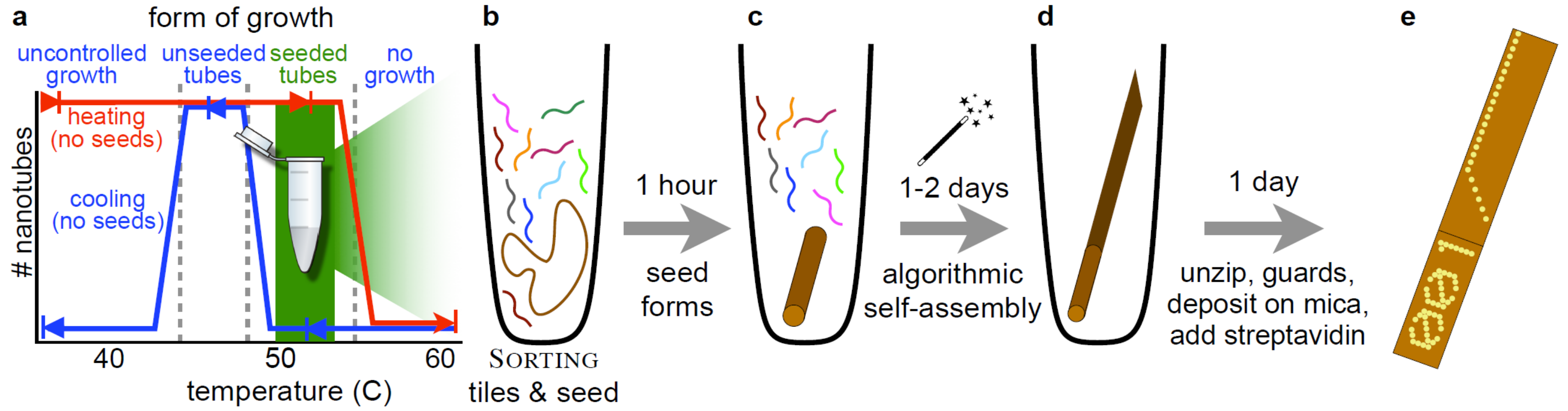


To execute circuit γ on input $x \in \{0,1\}^*$:

- Mix
  - origami (bar-coded to identify both γ and x)

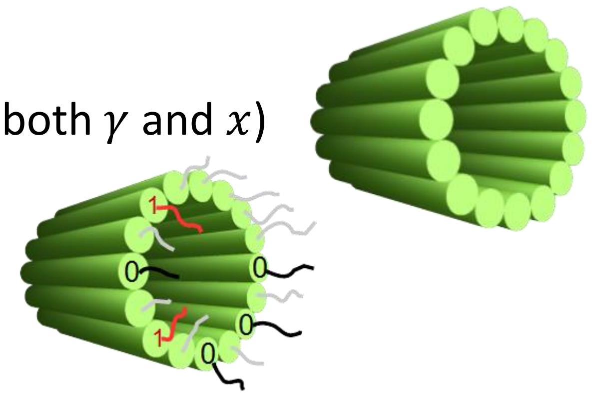


Experimental protocol

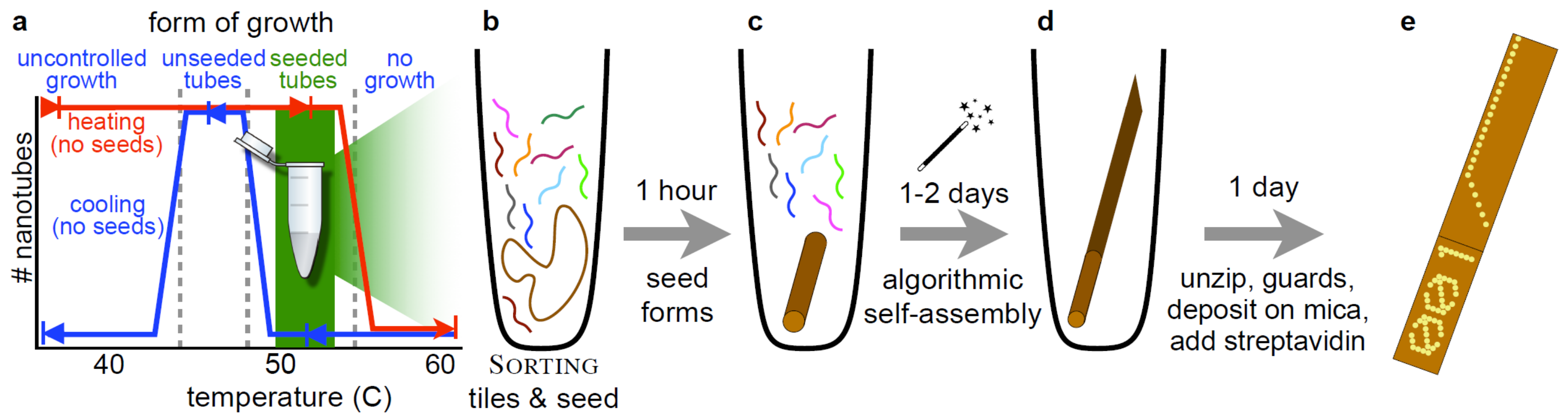


To execute circuit γ on input $x \in \{0,1\}^*$:

- Mix
    - origami (bar-coded to identify both γ and x)
    - "adapter" strands encoding x

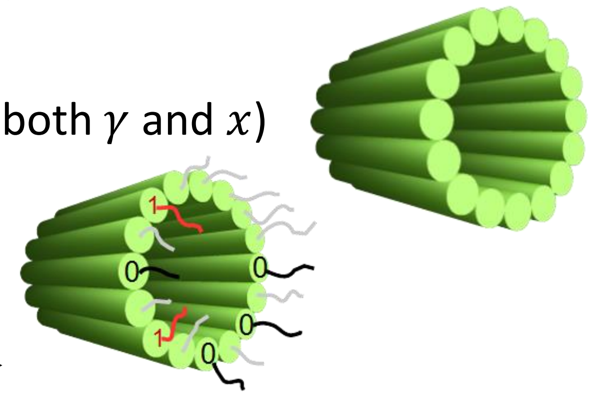


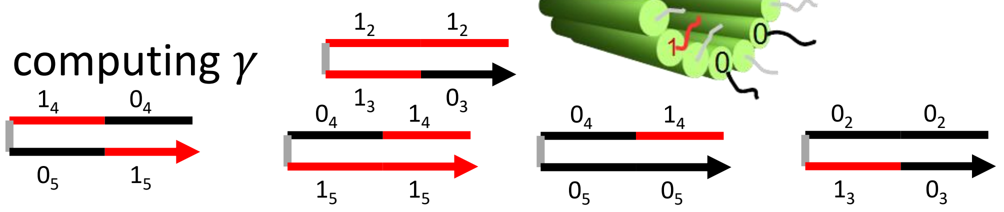
Experimental protocol



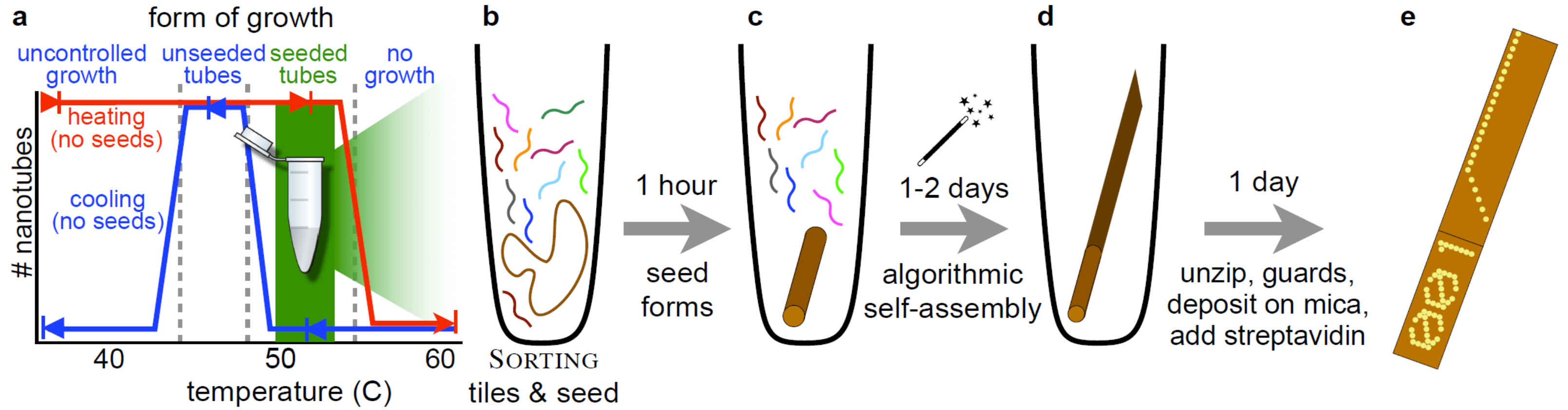
To execute circuit γ on input $x \in \{0,1\}^*$:

- Mix
  - origami (bar-coded to identify both γ and x)
  - “adapter” strands encoding x
  - tiles computing γ



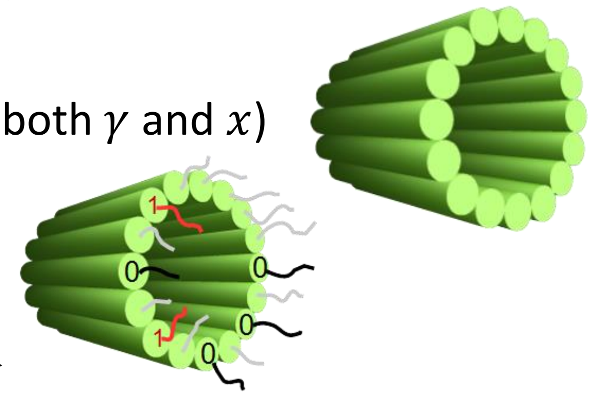


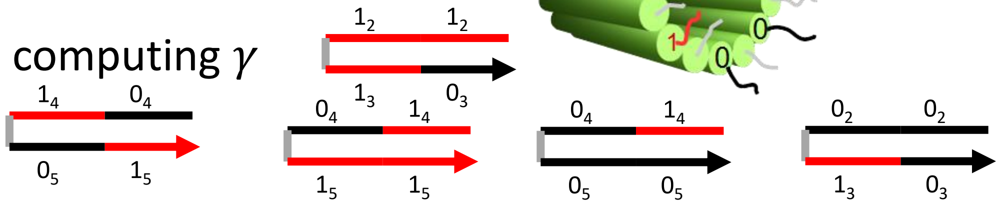
Experimental protocol



To execute circuit γ on input $x \in \{0,1\}^*$:

- Mix
  - origami (bar-coded to identify both γ and x)
  - “adapter” strands encoding x
  - tiles computing γ



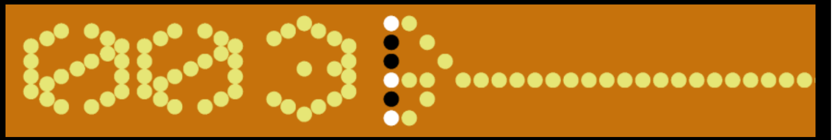


- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add “unzipper” strands (remove seam to convert tube to ribbon)
- Add “guard” strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

Results

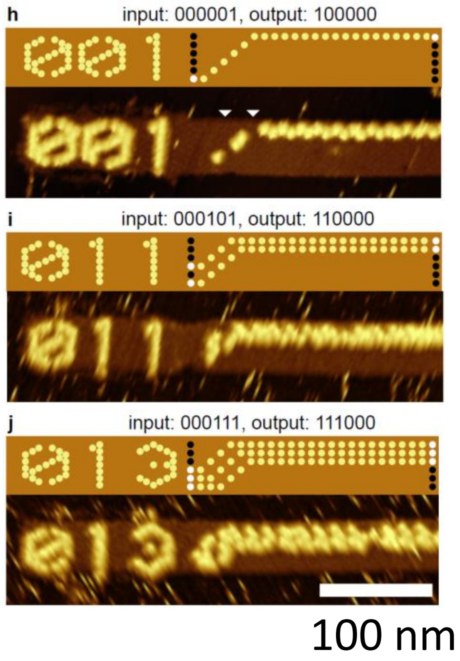
```
def test_parity():
    actual = parity('100101')

    expected =
```



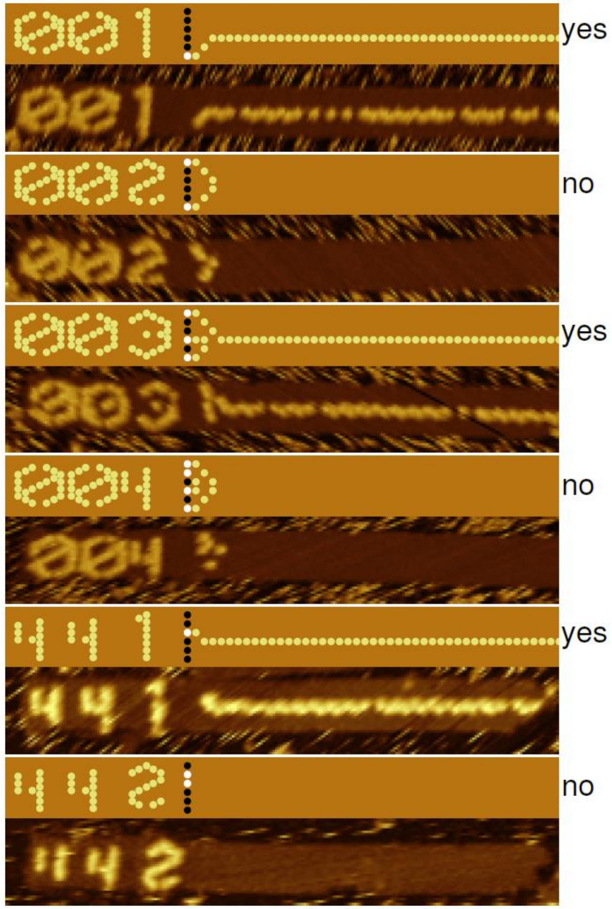
```
    assertEquals(expected, actual)
```

SORTING



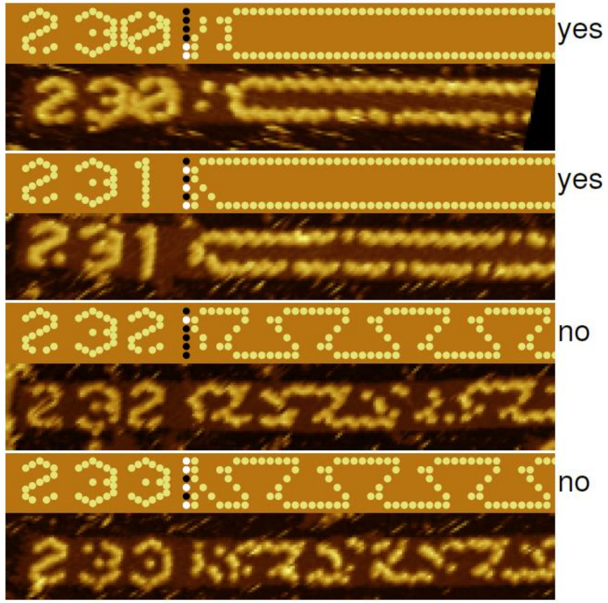
PARITY

Is the number of 1's odd?



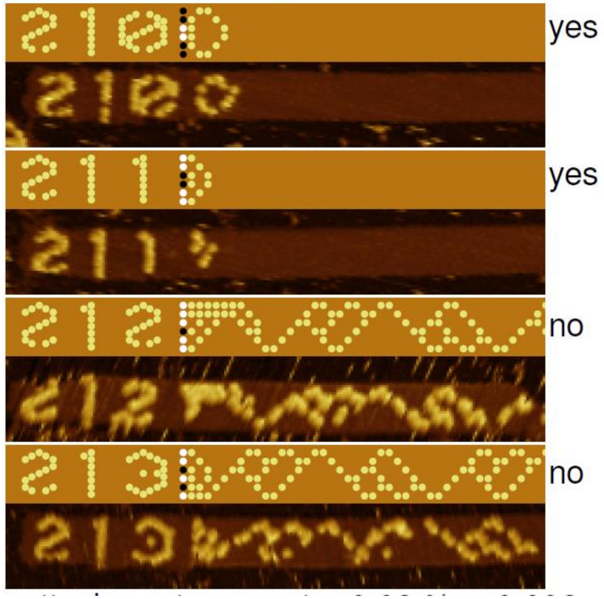
MULTIPLEOF3

Is the input binary number a multiple of 3?

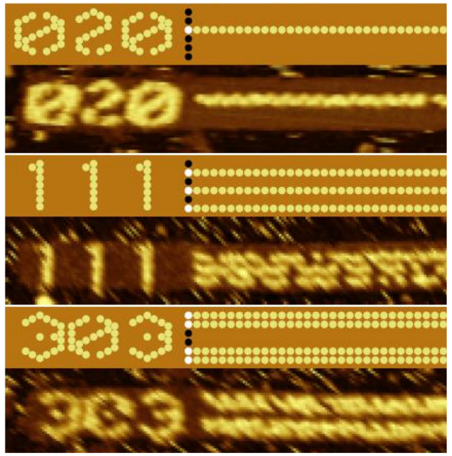


PALINDROME

Is the input a palindrome?

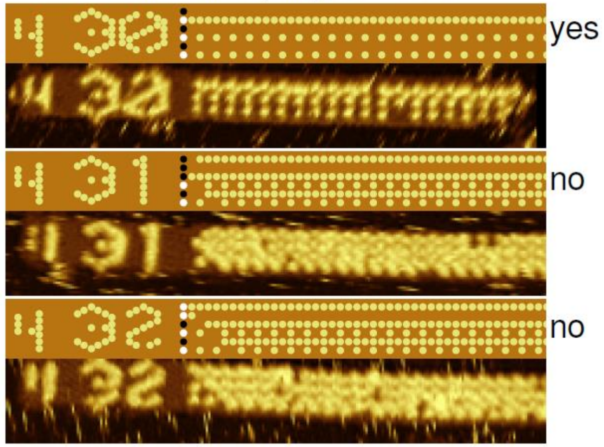


COPY



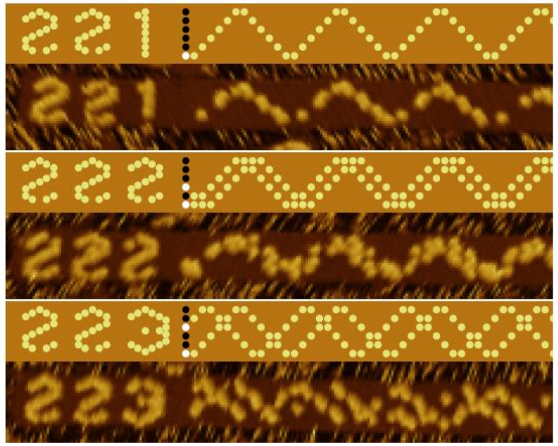
RECOGNISE21

Is the binary input $= 21$?

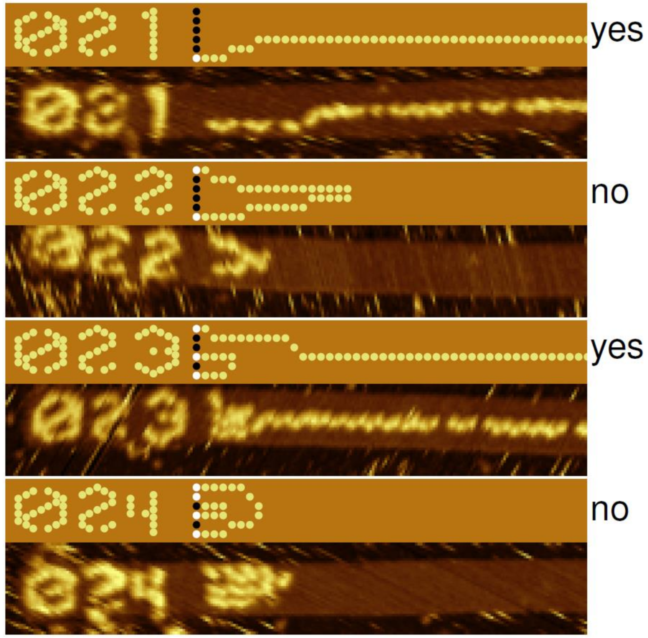


ZIG-ZAG

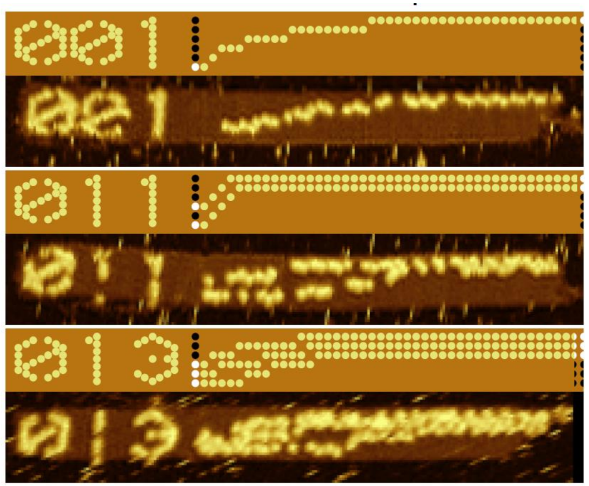
Repeating pattern



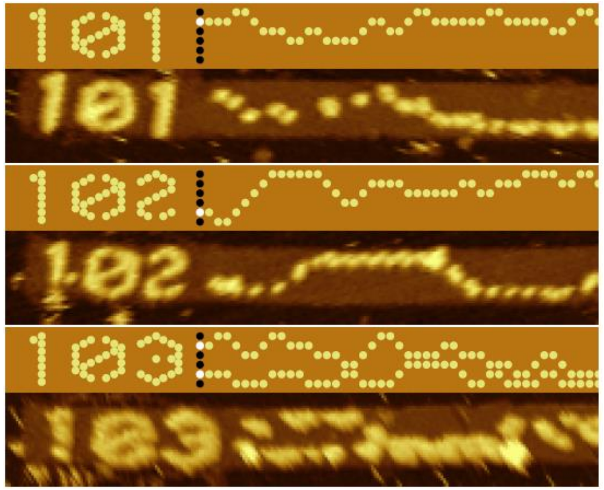
LazyParity



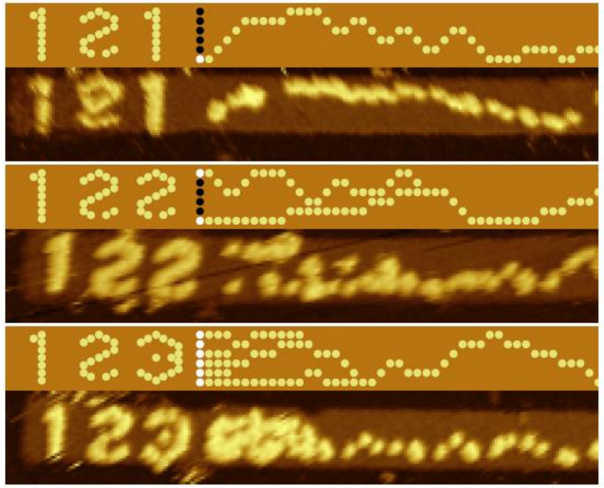
LazySorting



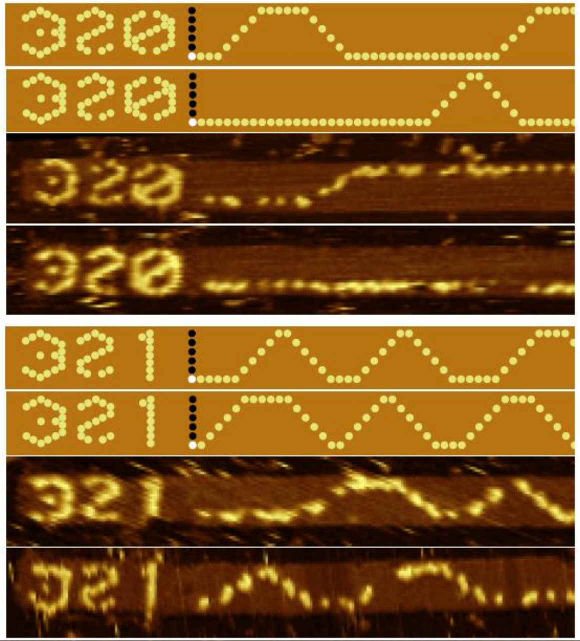
RandomWalkingBit



LeaderElection

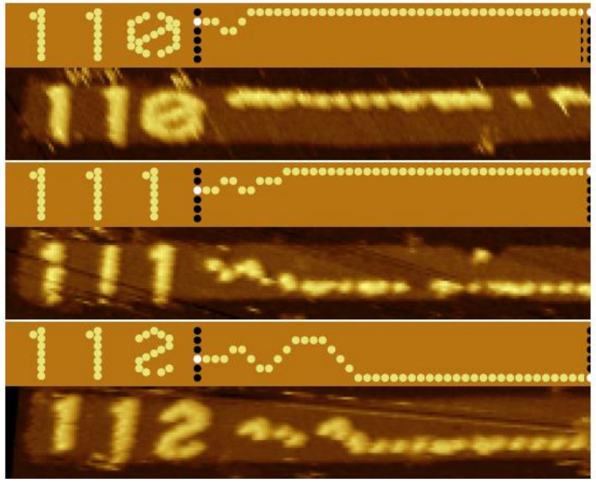


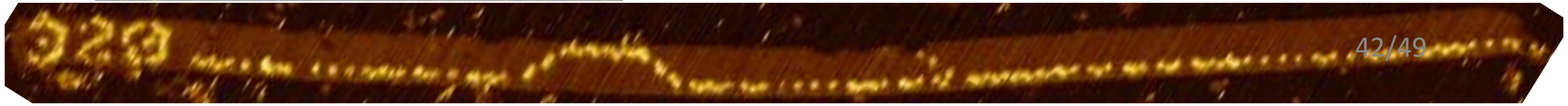
Waves



AbsorbingRandomWalkingBit

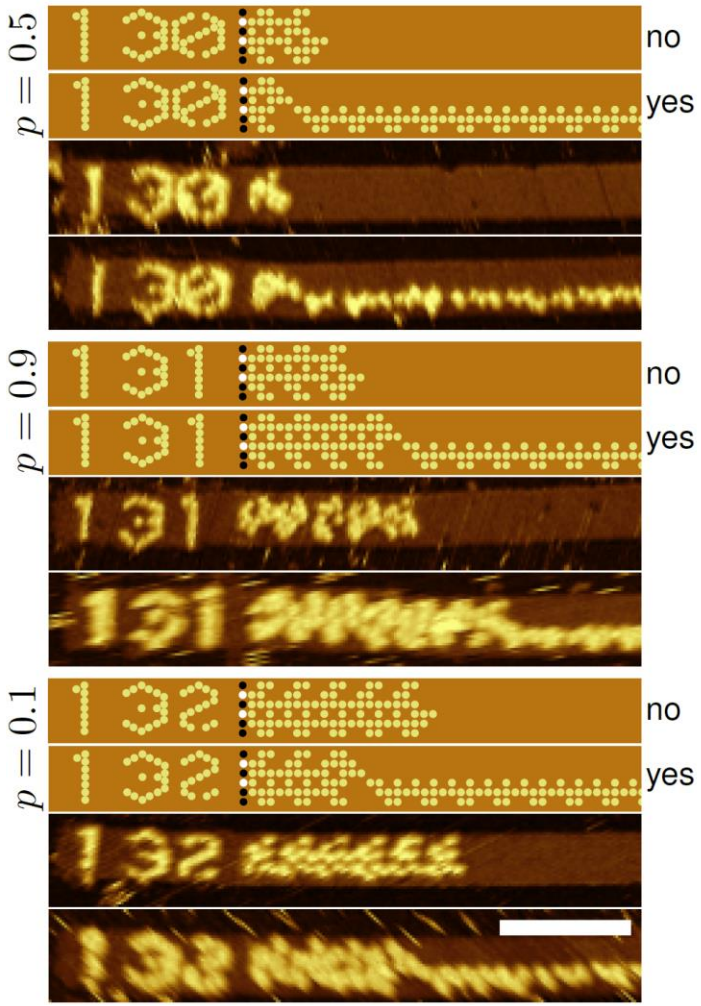
Random walker absorbs to top/bottom

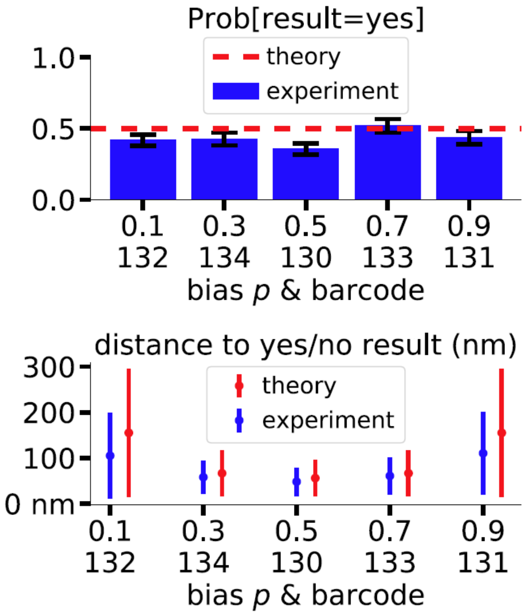




FairCoin

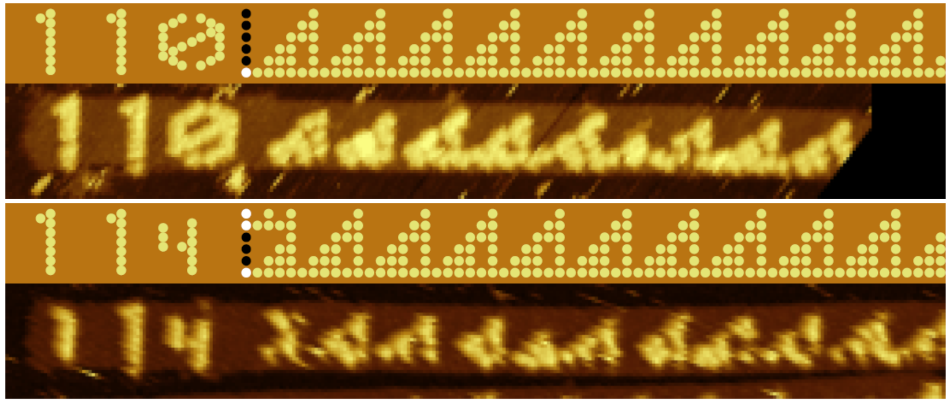
Unbiasing a biased coin





Rule110

Simulation of a cellular automaton



Counting to 63

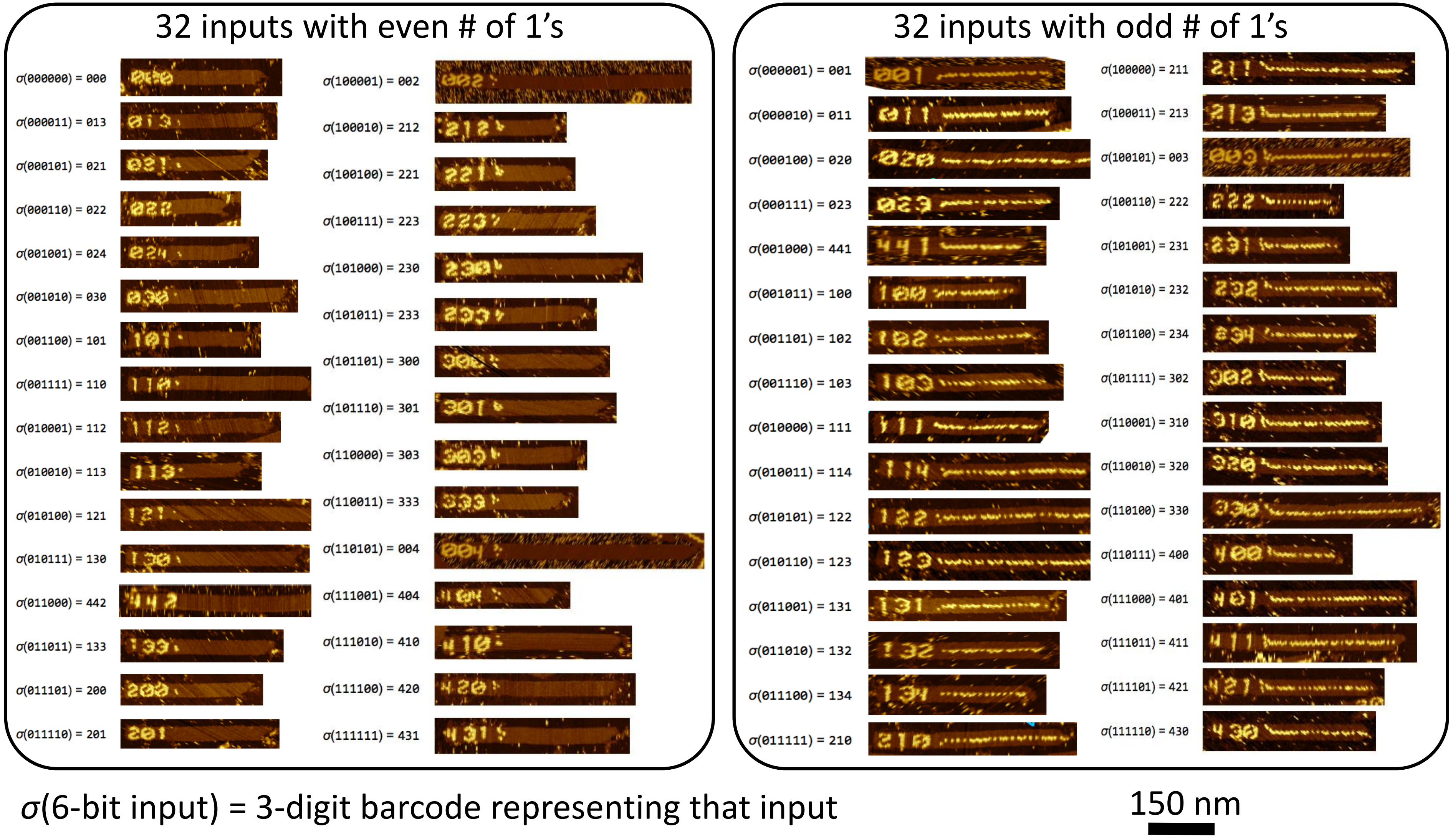
Circuit with 63 distinct strings

Is there a 64-counter?

No!
Proof by Tristan Stérin, ENS Lyon & Inria
(Consequence of following theorem: *Any bijective Boolean circuit having one output bit that does not depend on all of its input bits cannot compute odd bijections*.)

Parity tested on all inputs

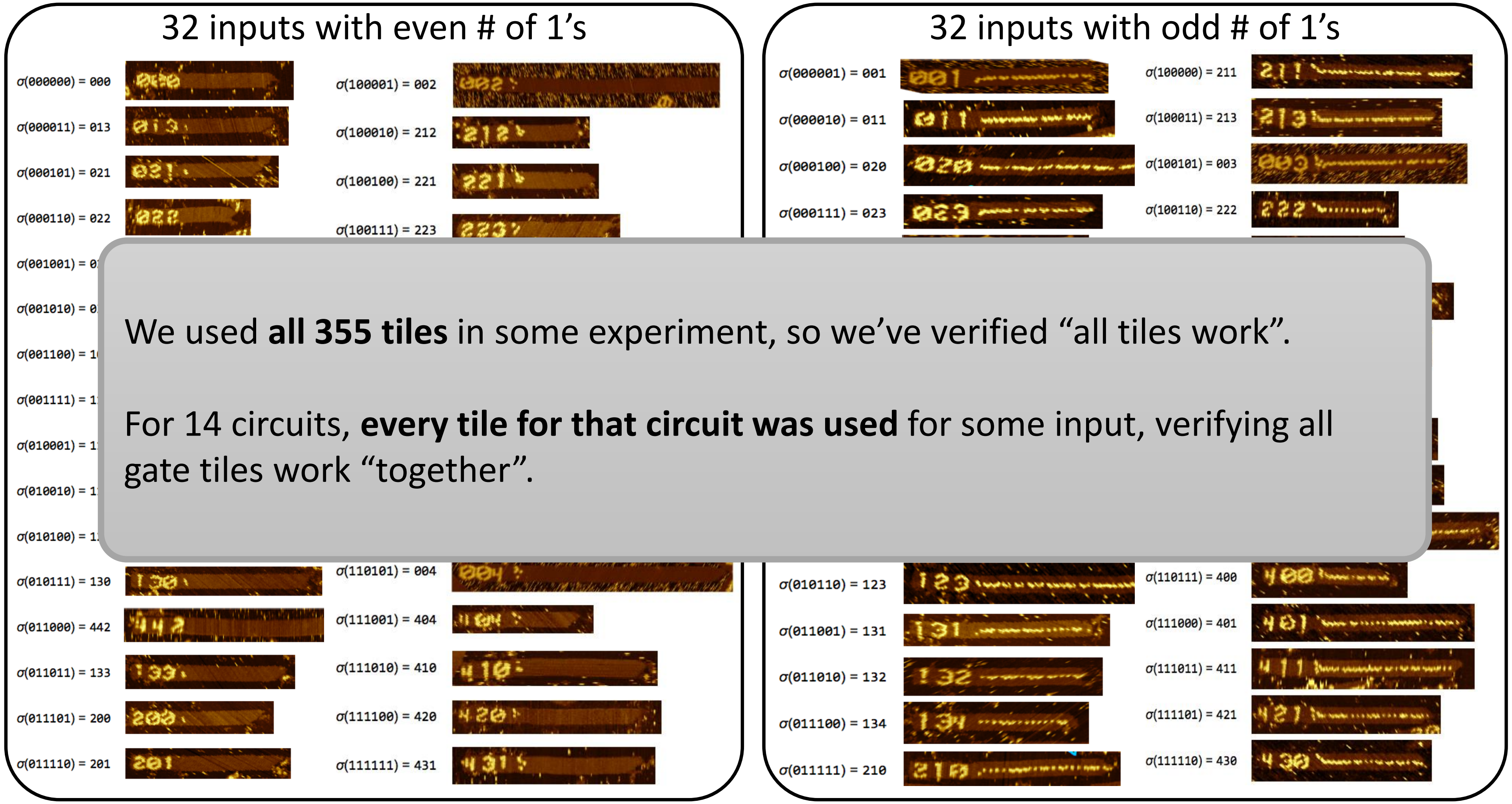
2^6 = 64 inputs with 6 bits



σ(6-bit input) = 3-digit barcode representing that input

Parity tested on all inputs

$2^6 = 64$ inputs with 6 bits



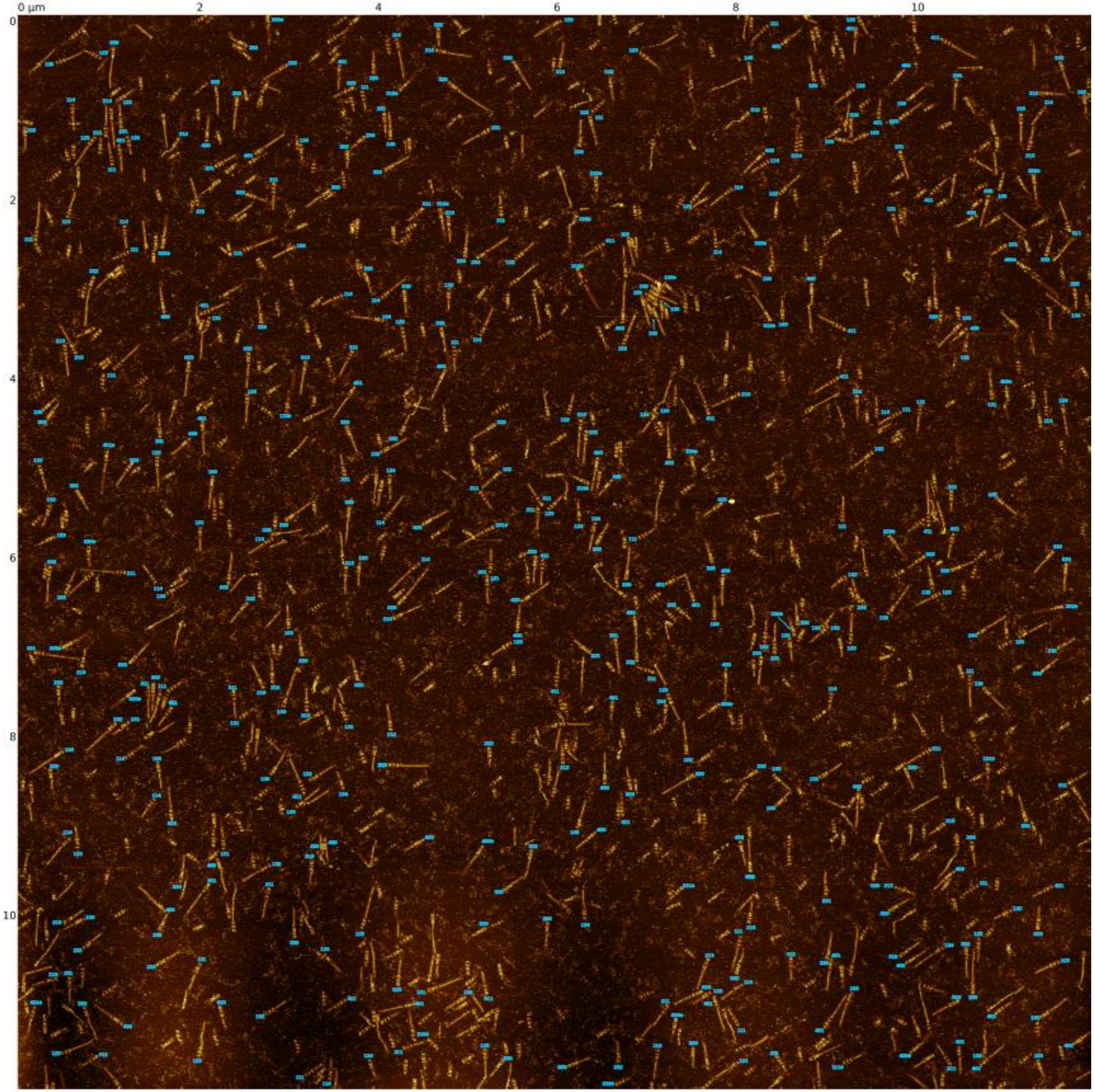
We used **all 355 tiles** in some experiment, so we've verified "all tiles work".

For 14 circuits, **every tile for that circuit was used** for some input, verifying all gate tiles work "together".

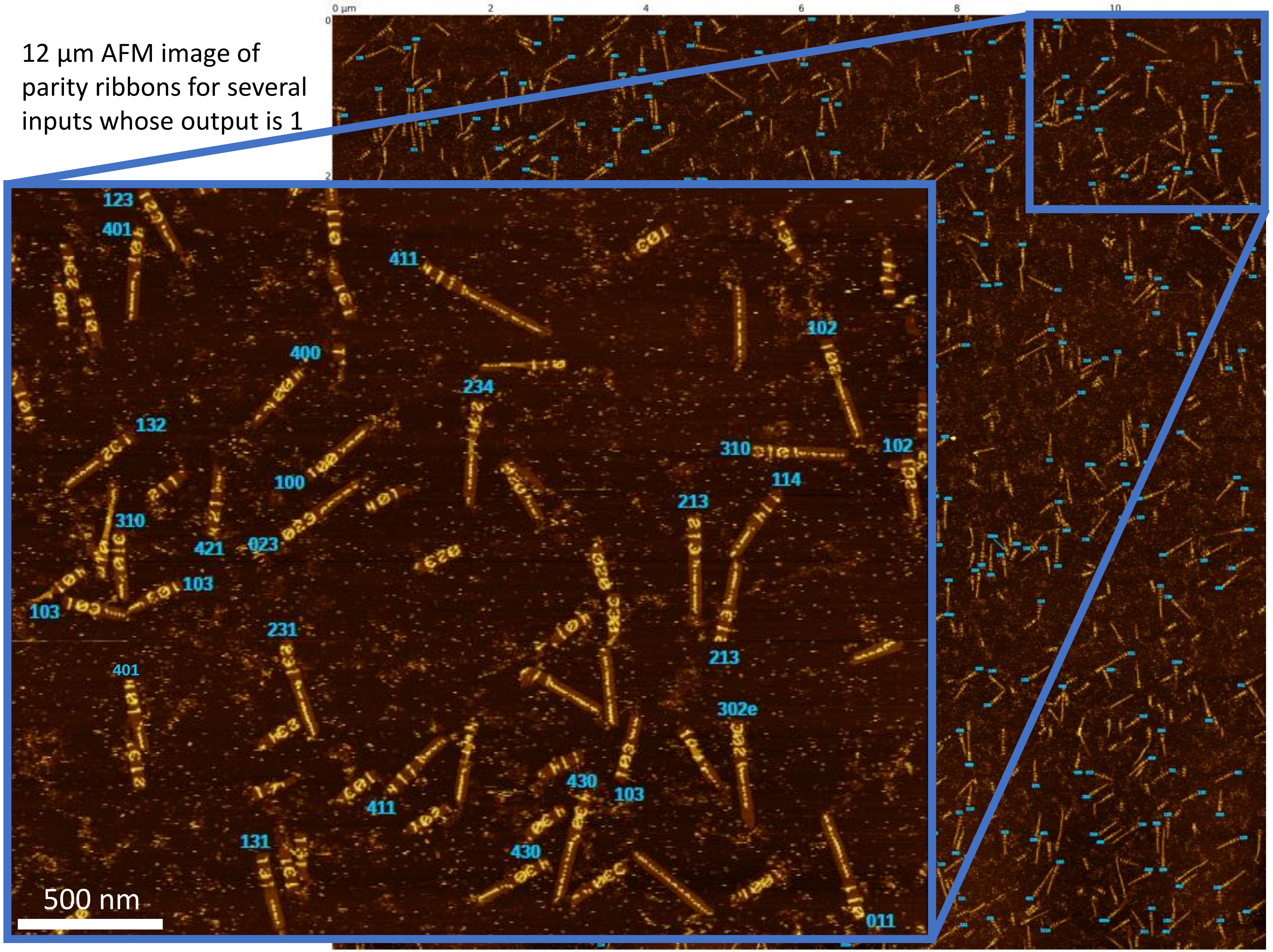
σ(6-bit input) = 3-digit barcode representing that input

150 nm

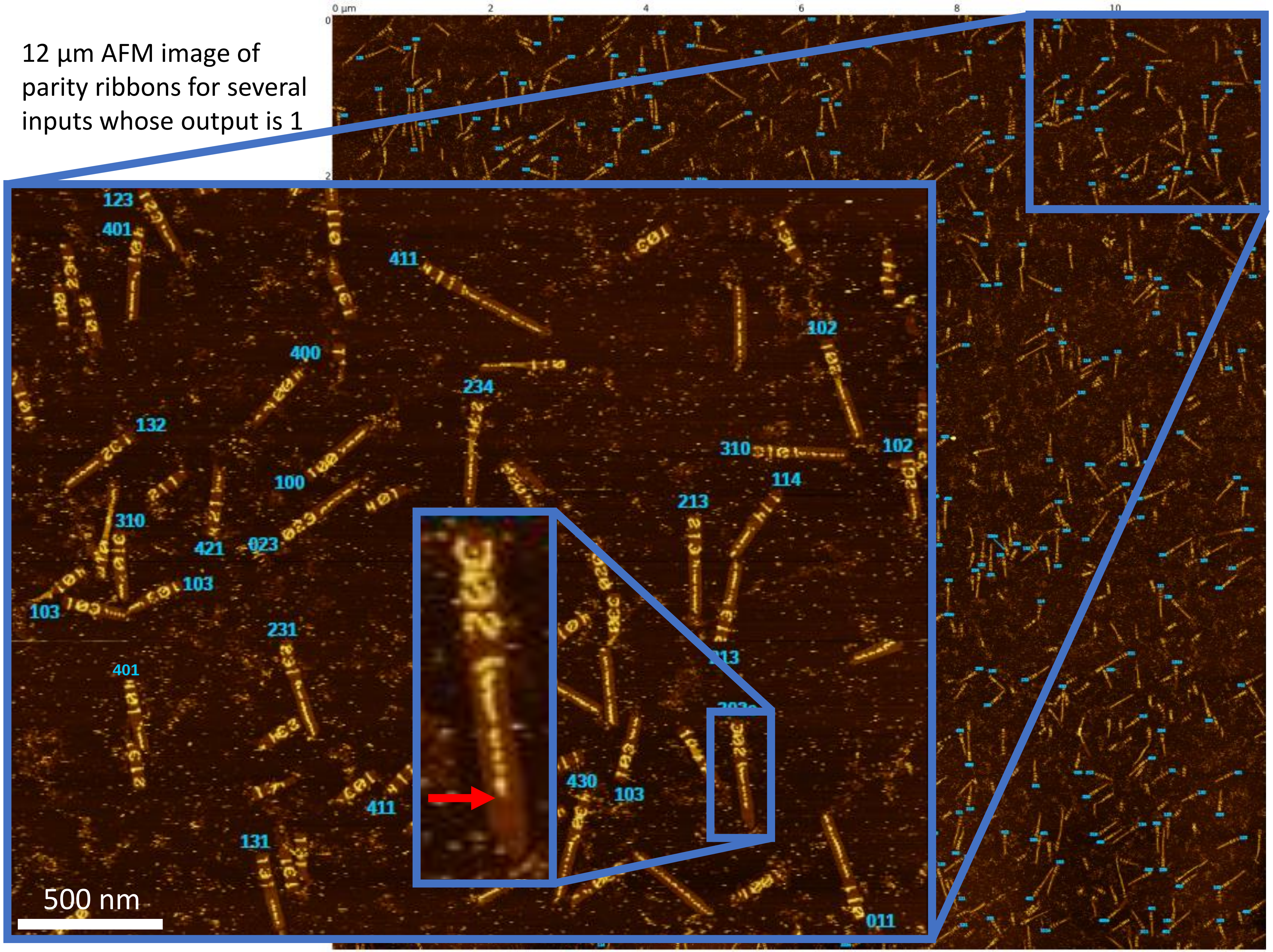
12 µm AFM image of parity ribbons for several inputs whose output is 1



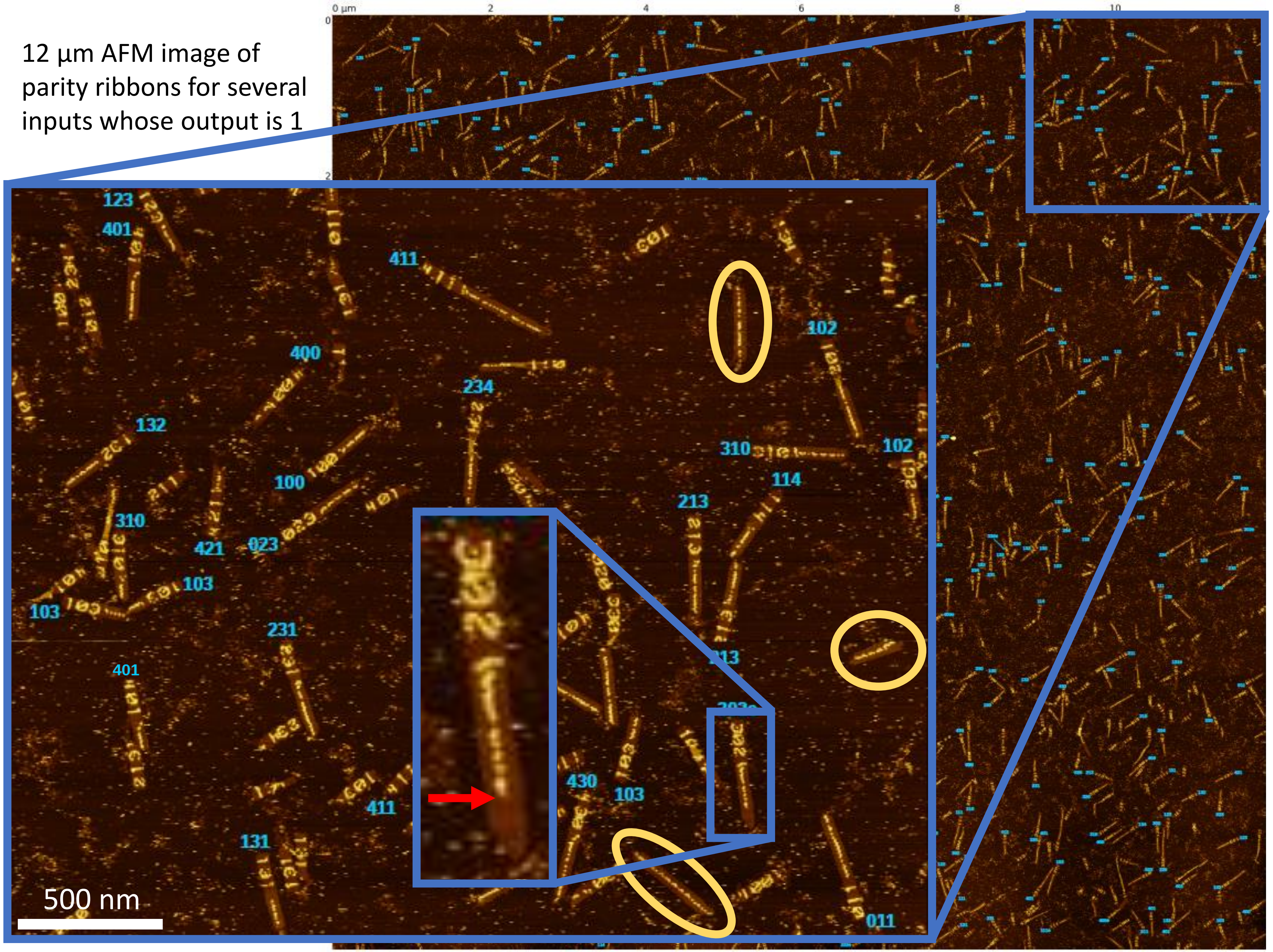
12 µm AFM image of parity ribbons for several inputs whose output is 1



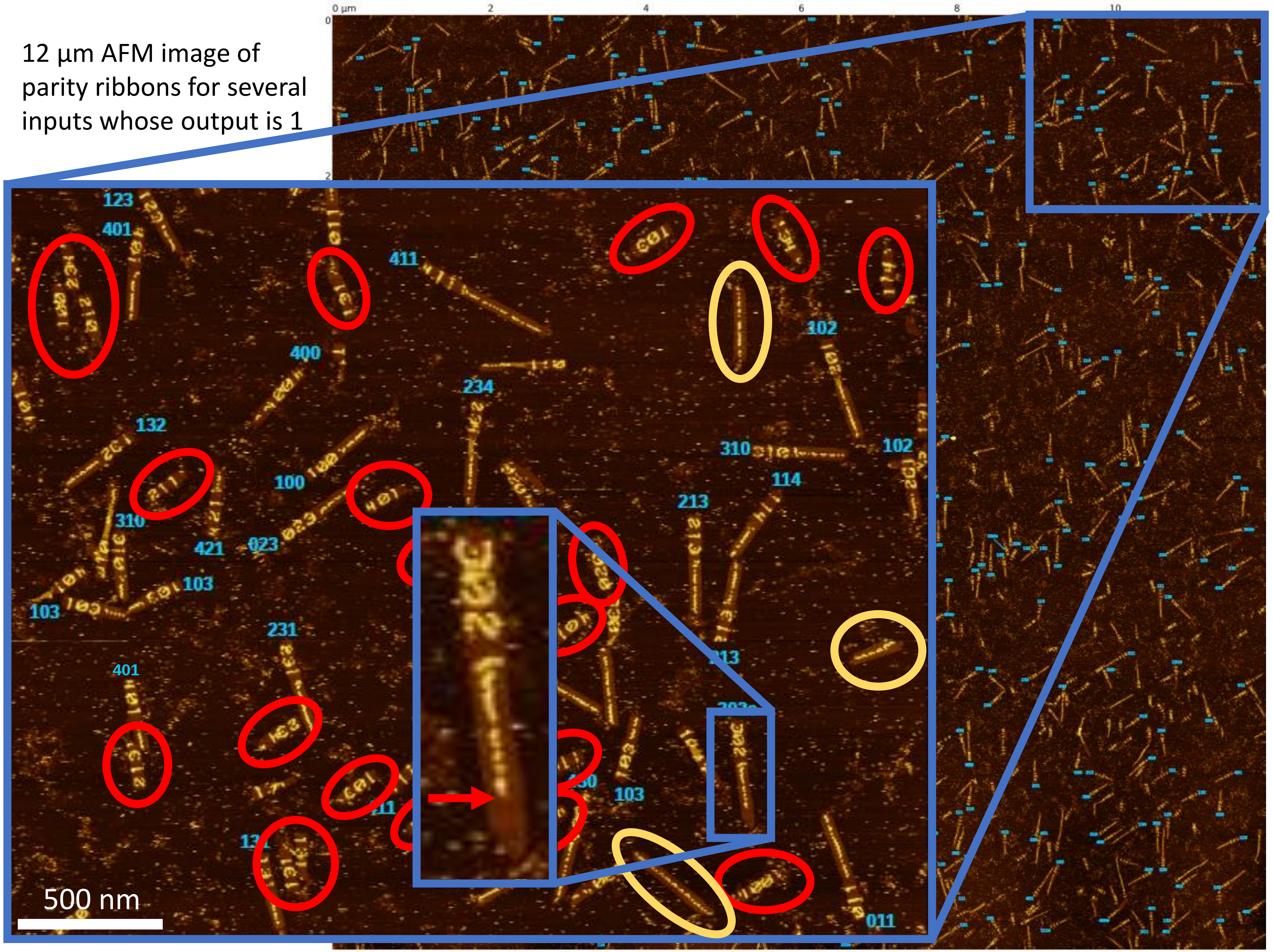
12 µm AFM image of parity ribbons for several inputs whose output is 1



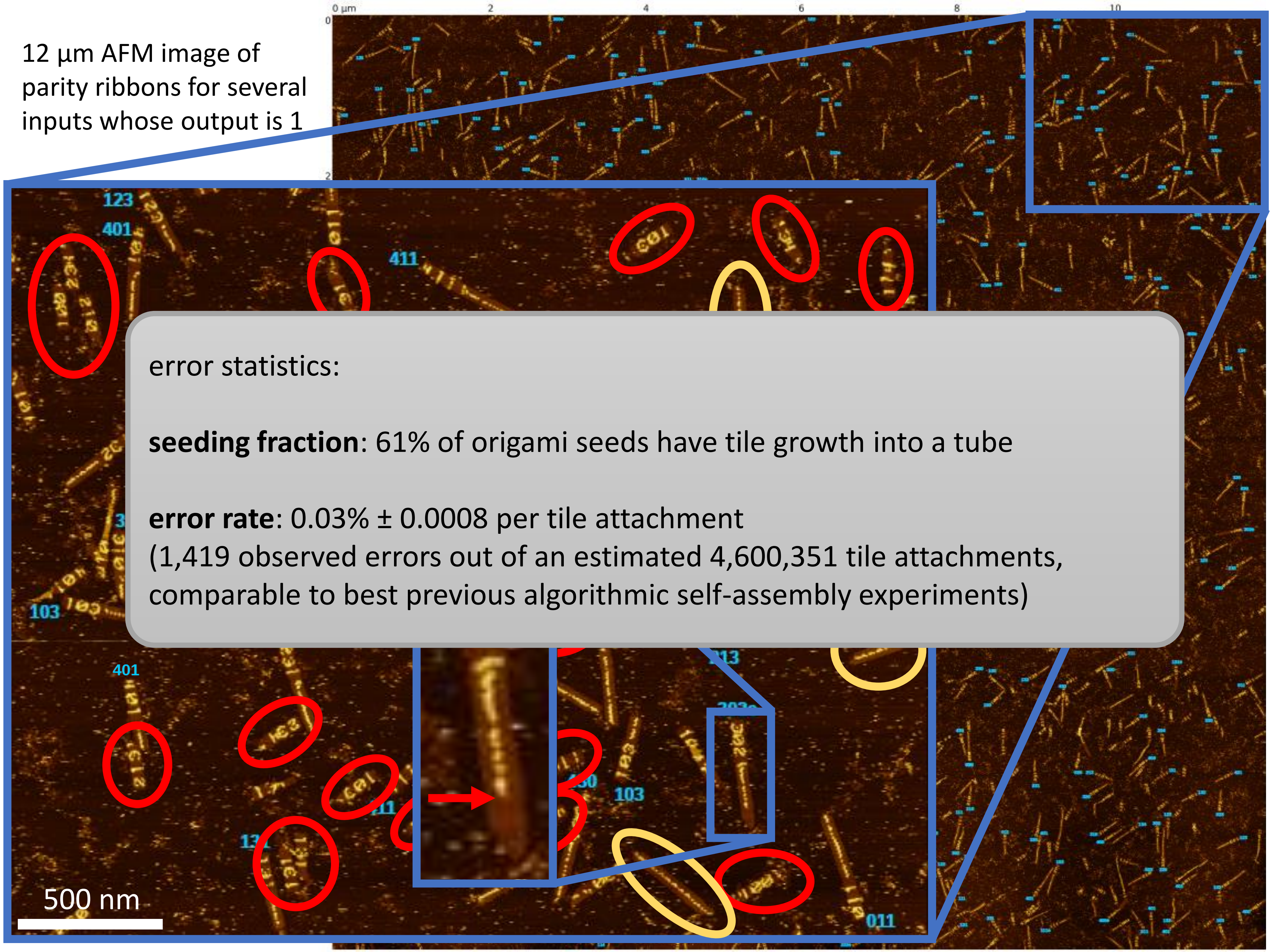
12 µm AFM image of parity ribbons for several inputs whose output is 1



12 µm AFM image of parity ribbons for several inputs whose output is 1



12 μm AFM image of parity ribbons for several inputs whose output is 1



error statistics:

seeding fraction: 61% of origami seeds have tile growth into a tube

error rate: 0.03% ± 0.0008 per tile attachment
(1,419 observed errors out of an estimated 4,600,351 tile attachments, comparable to best previous algorithmic self-assembly experiments)

What did we learn?

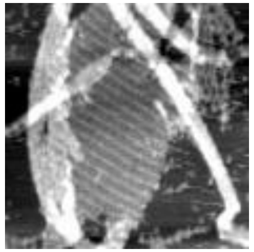
A <u>small</u>(ish) library of molecules can be <u>reprogrammed</u> to self-assemble <u>reliably</u> into many complex patterns, by <u>processing information</u> as they grow.

What did we learn?

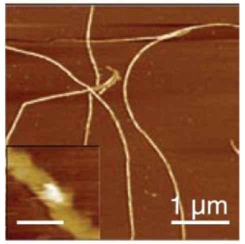
A small(ish) library of molecules can be reprogrammed to self-assemble reliably into many complex patterns, by processing information as they grow.

Contrasting with other self-assembly work:

more algorithmic control than **periodic** self-assembly



double-crossover tile lattices (Winfree et al., *Nature* 1998)



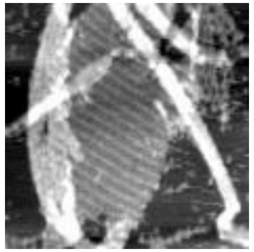
single-stranded tile tubes (Yin et al., *Science* 2008)

What did we learn?

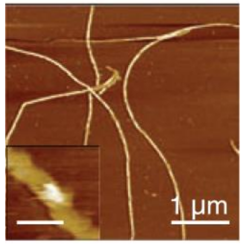
A small(ish) library of molecules can be reprogrammed to self-assemble reliably into many complex patterns, by processing information as they grow.

Contrasting with other self-assembly work:

more algorithmic control than **periodic** self-assembly

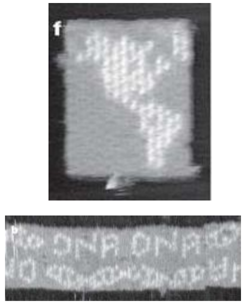


double-crossover tile lattices (Winfree et al., *Nature* 1998)



single-stranded tile tubes (Yin et al., *Science* 2008)

fewer types of DNA strands required than **uniquely-addressed** self-assembly



DNA origami (Rothemund, *Nature* 2006)

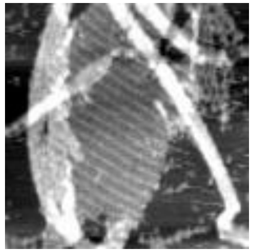
single-stranded tile lattice (Wei et al., *Nature* 2012)

What did we learn?

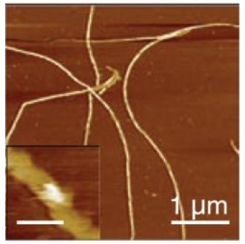
A small(ish) library of molecules can be reprogrammed to self-assemble reliably into many complex patterns, by processing information as they grow.

Contrasting with other self-assembly work:

more algorithmic control than **periodic** self-assembly

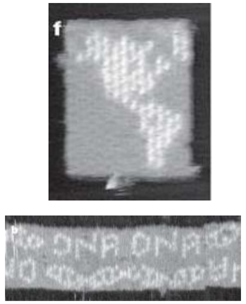


double-crossover tile lattices (Winfree et al., *Nature* 1998)



single-stranded tile tubes (Yin et al., *Science* 2008)

fewer types of DNA strands required than **uniquely-addressed** self-assembly

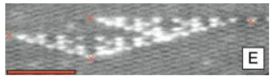


DNA origami (Rothemund, *Nature* 2006)

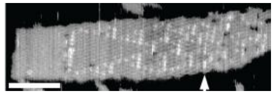
single-stranded tile lattice (Wei et al., *Nature* 2012)

order of magnitude more tile types available than previous **algorithmic** self-assembly

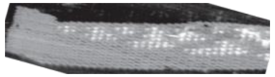
double-crossover tile lattices



(Rothemund et al., *PLoS Bio* 2004)



(Fujibayashi et al., *Nano Letters* 2008)



(Barish et al., *PNAS* 2009)

(Evans, *Ph.D. thesis* 2014)

Next big challenge: Algorithmically control shape

We “drew” interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Next big challenge: Algorithmically control shape

We “drew” interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S “encoding” S, T self-assembles S.

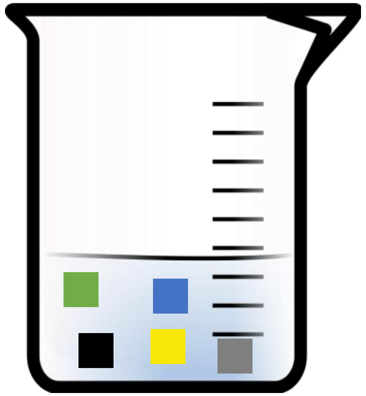
[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



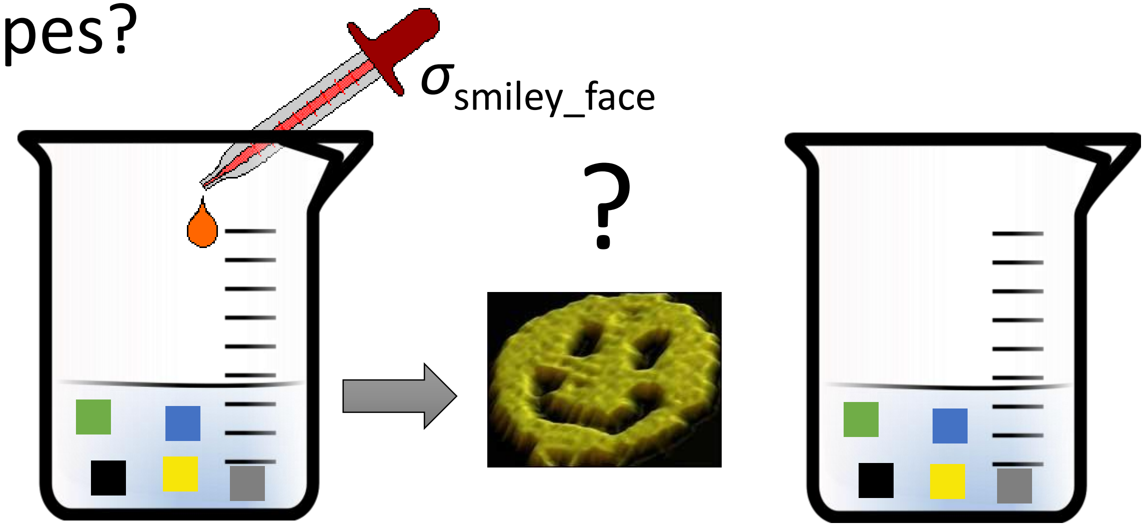
[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



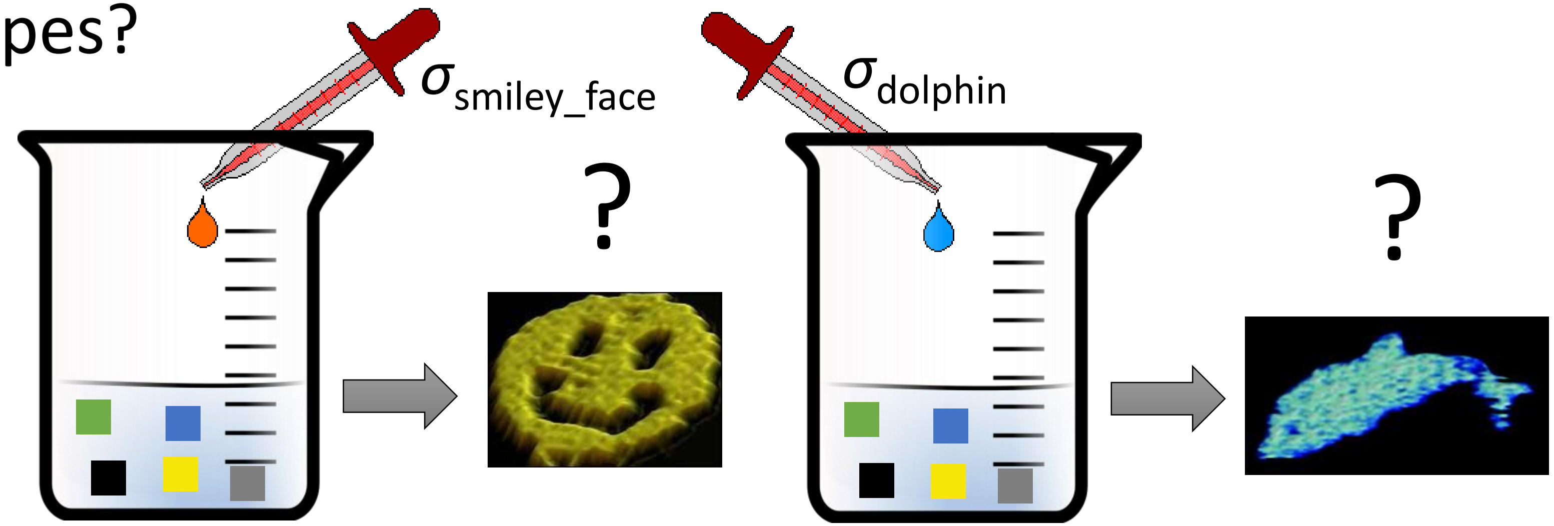
[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



These tiles are universally programmable for building any shape.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Thank you!