CS294-7: Reconfigurable Computing

Day 10: Thursday, February 20

Spatial Computing Styles

- Admin
- Cellular Automata

Admin

- CAD Tools for Project 1

Cellular Automata

Cellular Automata: Computing structure composed of a lattice of simple, synchronous cells with finite state interacting locally.

- **homogeneous array**
- **synchronous**
- **FSM**
  - **finite and discrete state**
  - **deterministic update**
  - **temporal locality**
- **spatial locality (inputs only from local neighborhood)**

Initially, logical model for describing computations.

CA Use

- Direct models of systems with local interactions
  - crystal growth
  - chemical reaction/diffusion
  - molecular dynamics
  - fluid flow
- Image processing
  - pattern recognition
  - erosion/dilation
  - feature extraction
- CAD
  - wire routing (path finding)
  - DRC (image processing)

LIFE

Canonical, toy example of a cellular automata

Each cell is in one of two states: live or dead

On each cycle, the state of a cell is updated based on its neighbors.

- **Generation** – empty cell with 3 live neighbors becomes live
- **Isolation** – live cell with $\leq$ 1 neighbor dies
- **Survival** – live cell with 2 or 3 neighbors lives
- **Overpopulation** – live cell with $\geq$ 4 neighbors dies

Maze Routing

Task: Find least expensive path between a source, S, and a target, T, given obstacles.
(obstacles may include previously routed wires.)

Lee's Algorithm:

- Perform a breadth first search (by cost) from source.
- Note backlink to previous link
- Reconstruct path from backlinks once find target
- n – cost/length of path
  $O(n^2)$ operations

Lee's Algorithm (example)

S	$\otimes$	$\otimes$		
		$\otimes$	$\otimes$	T

$\downarrow$				
S	$\otimes$	$\otimes$		
$\uparrow$		$\otimes$	$\otimes$	T

$\downarrow$	$\leftarrow$			
S	$\otimes$	$\otimes$		
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T
$\uparrow$				

$\downarrow$	$\leftarrow$	$\leftarrow$		
S	$\otimes$	$\otimes$		
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T
$\uparrow$	$\leftarrow$			
$\uparrow$				

$\downarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	
S	$\otimes$	$\otimes$		
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T
$\uparrow$	$\leftarrow$	$\leftarrow$		
$\uparrow$	$\leftarrow$			

$\downarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
S	$\otimes$	$\otimes$	$\uparrow$	
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	
$\uparrow$	$\leftarrow$	$\leftarrow$		

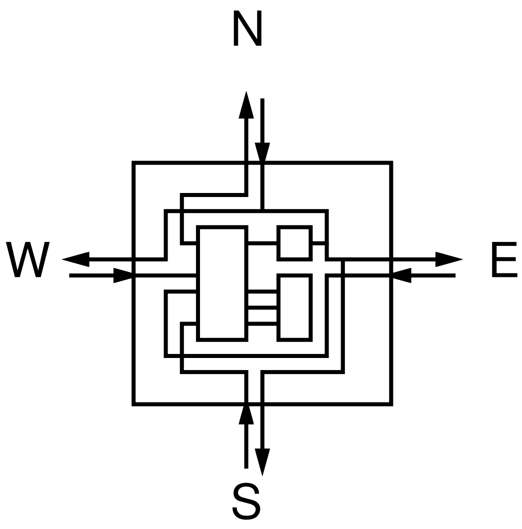
$\downarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
S	$\otimes$	$\otimes$	$\uparrow$	$\uparrow$
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	

$\downarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
S	$\otimes$	$\otimes$	$\uparrow$	$\uparrow$
$\uparrow$	$\leftarrow$	$\otimes$	$\otimes$	T $\uparrow$
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$

Maze Search CA

encode "cost" in arrival time

```
if (not(obstacle))
  if (westin) dir=west; out=1
  else if (northin) dir=north; out=1
  else if (eastin) dir=east; out=1
  else if (southin) dir=south; out=1
  else out=0
else out=0
```



- shift in image
- route completion detection

Maze Search CA – Tweaks

- Multilayer routing
  - add via direction
  - interleave cells for various layers
  - varying preference(cost)/layer

- Why not search from both ends?

Maze Search CA – time and decomposition

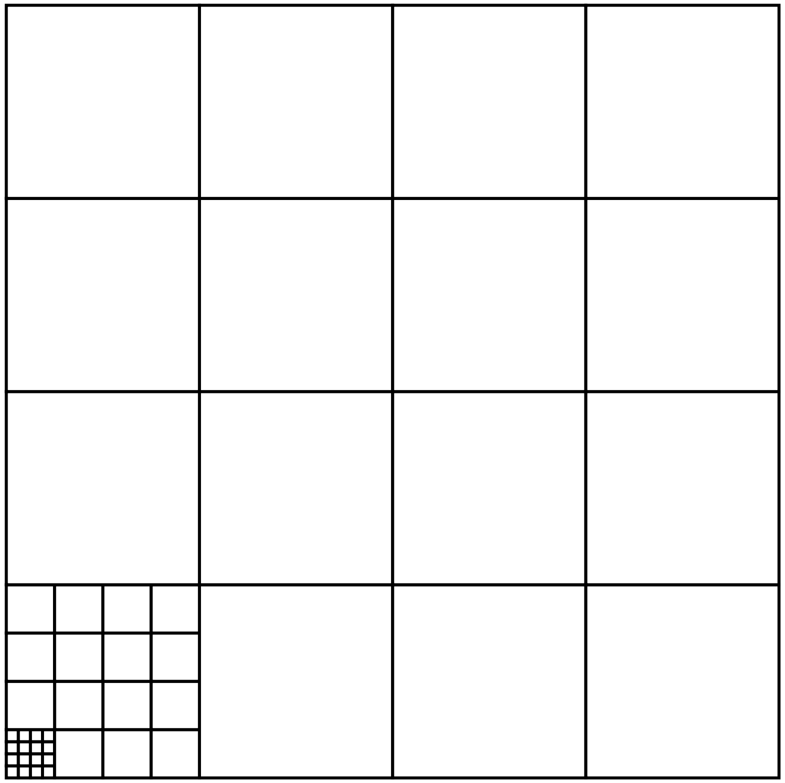
If a routing window fits,
takes $O(n)$ time to find route.

Otherwise, can only compute one step per window (need results which may be one step away across boundary).

1. multi-grid routing
2. raster pipeline

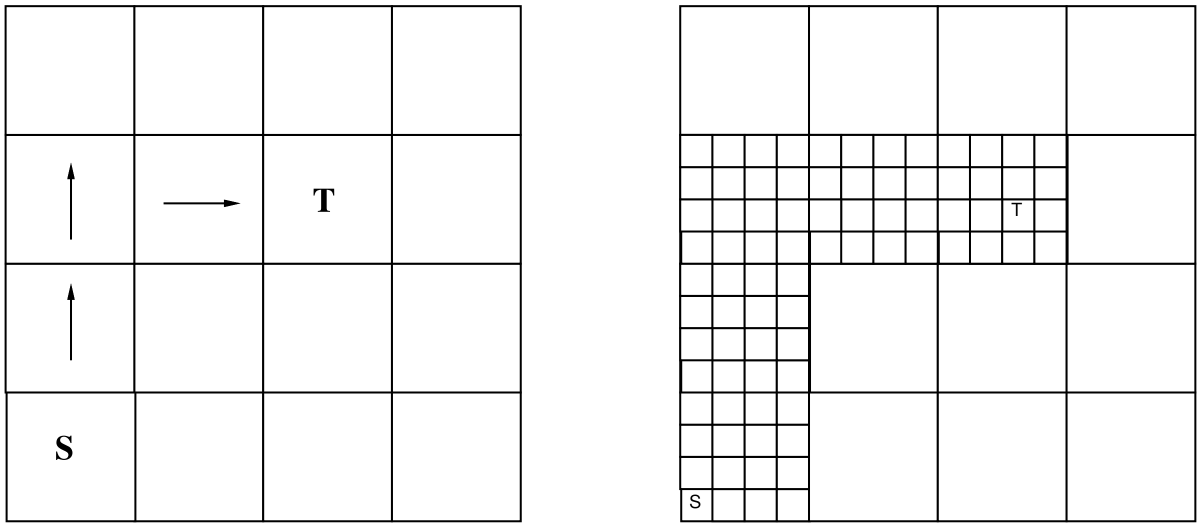
Maze Search CA – Multigrid

- **Divide into sequence of window sizes**
  (each cell now represents a $c \times c$ region of space)
  sequence $c = \frac{n}{w}$**,**$c = \frac{n}{kw}$**,** $c = \frac{n}{k^2 w}$**,**$\ldots$**,**$c = k$**,** $c = 1$



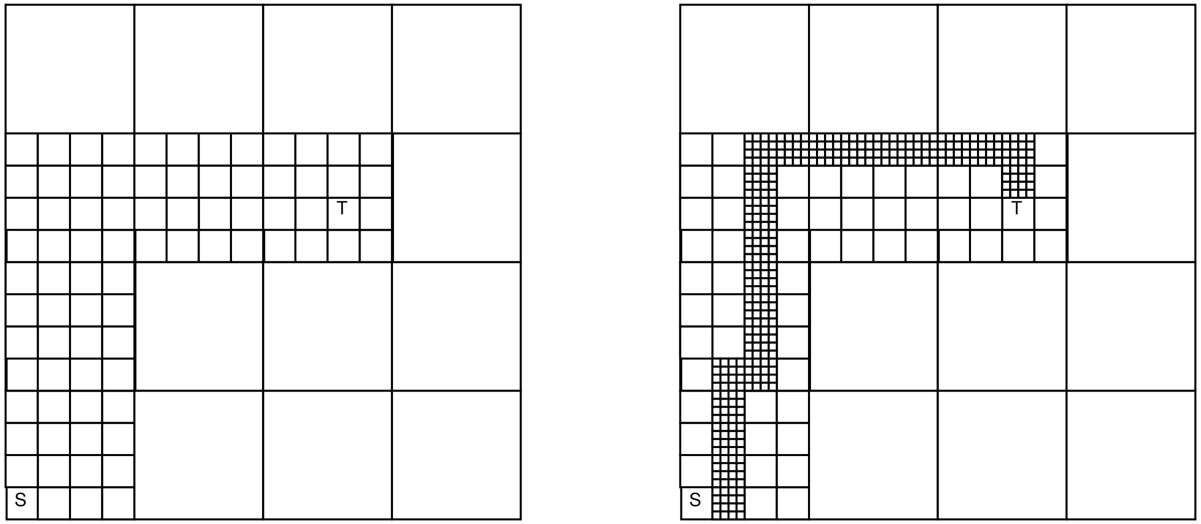
Maze Search CA – Multigrid

- Pick out path for next resolution
- Path limits and orders the windows must visit at next resolution



Maze Search CA – Multigrid

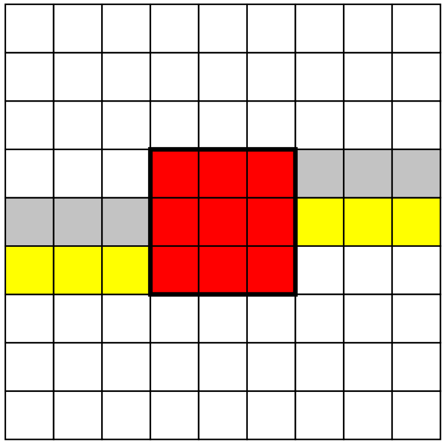
- **(probably end up using overlapping windows in practice to avoid edge effects)**
- **(higher levels push around summary information?)**
- $O(n)$ **time if pick out paths of size** $O(\frac{n}{c})$ **at each stage**

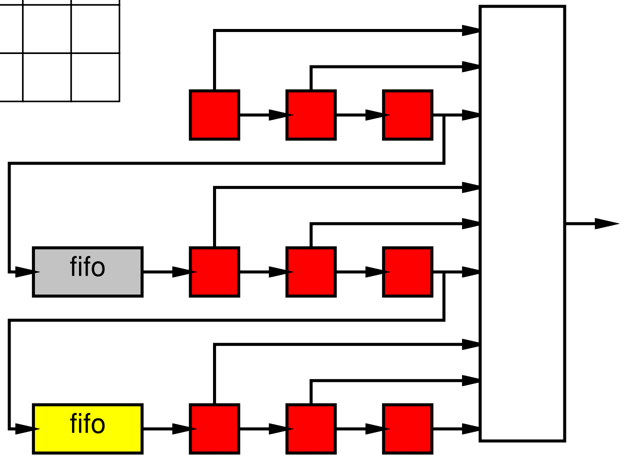


Common tradeoff in physical system CA – operate precisely on single-particles v/s operating statistically on particle aggregates.

Raster Processing of CA

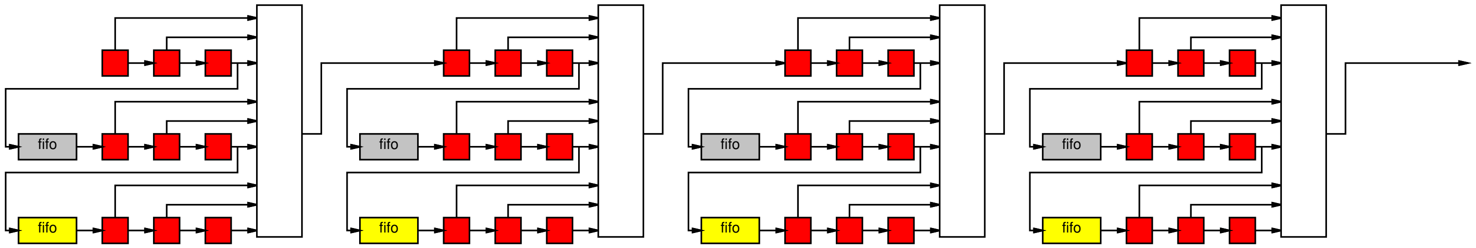
- To evaluate CA only need to see small window
- Serialize window-update evaluation
- As soon as finish one step on window, can start next step in pipelined fashion on downstream processor.





Raster Pipeline

- Cascade stages (up to E(Path Length))



Raster Pipeline

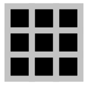
- Path length (l) stages (stage passes)
- $n \times n$ array $\rightarrow O(n^2 + n \cdot l)$
- Variable (expanding) window
  $\rightarrow$ focus work on active portion of array
  (most beneficial when reusing small number of hardware stages)

Design Rule Check

Tasks:

1. Connectivity Resolution – wavefront expansion and unification
2. Layer Combination – boolean combination
3. Tolerance Checks – image operations

Patterns (running examples)

$S(3) = Q(3) =$ **3×3 square** 

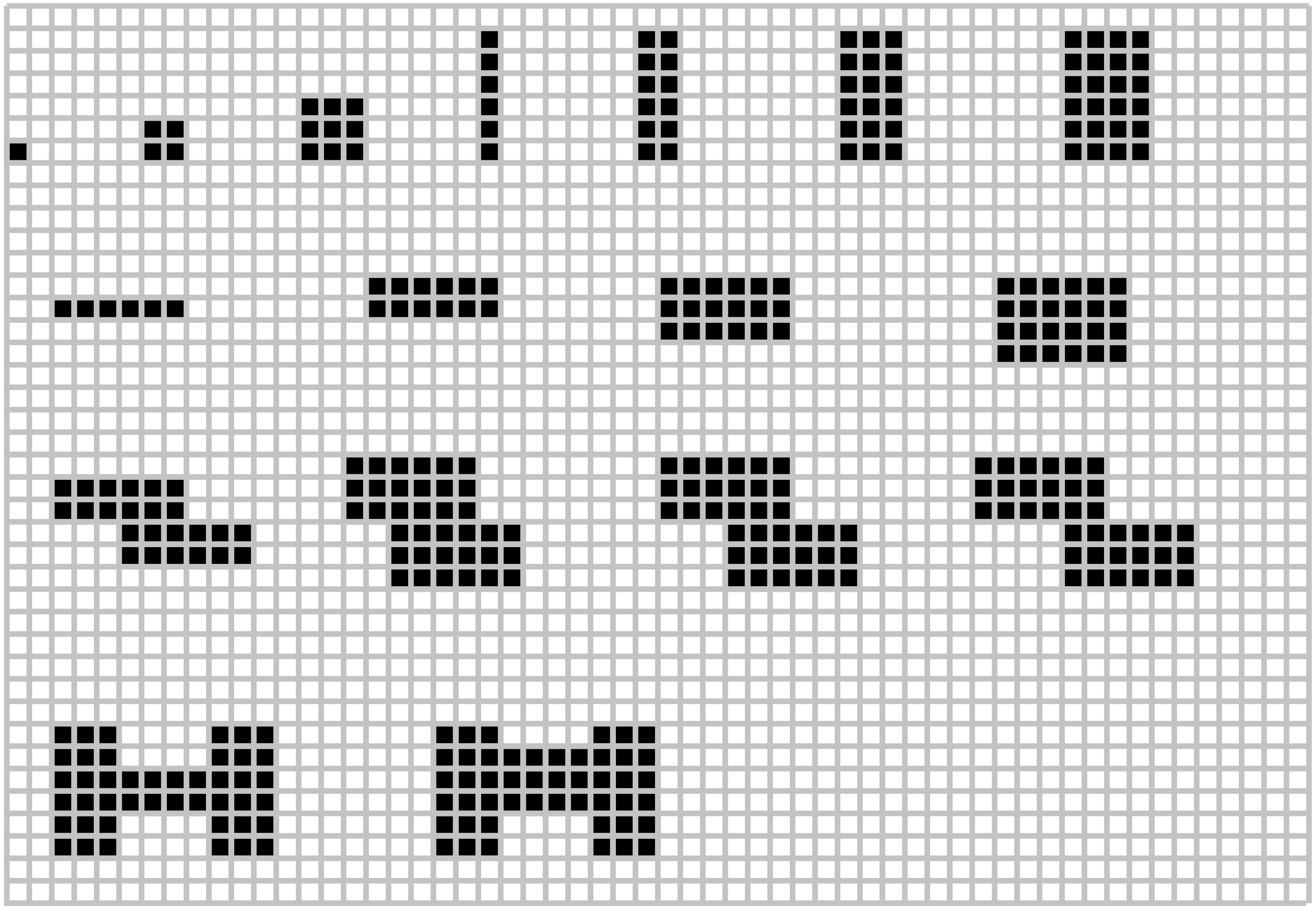


Image Operation – Dilation

Dilation: $A \oplus B$ – union of all points in A translated by all points in B.

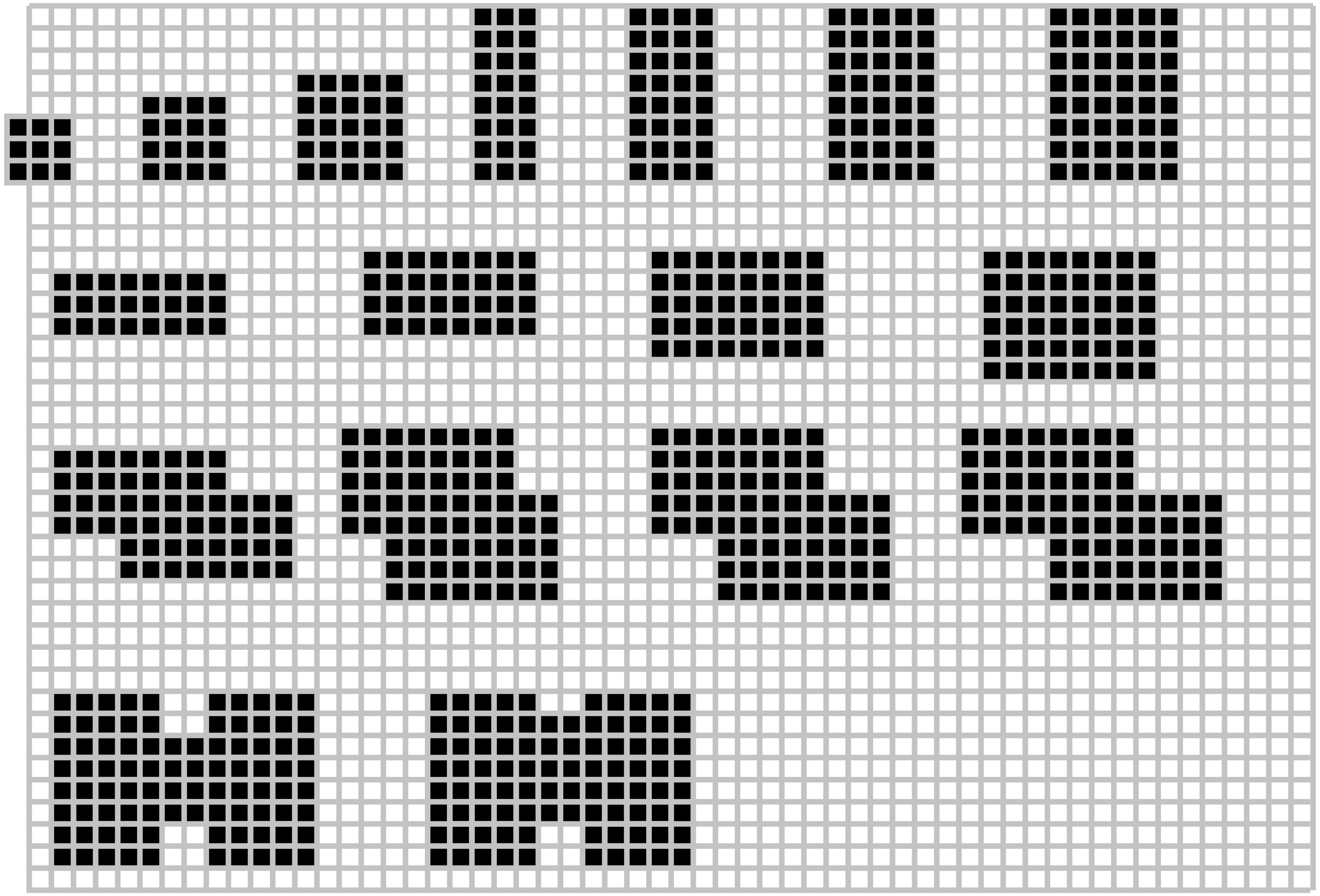


Image Operation – Erosion

Erosion: $A \ominus B$ – all points in A at which a translation of B still fits in A.

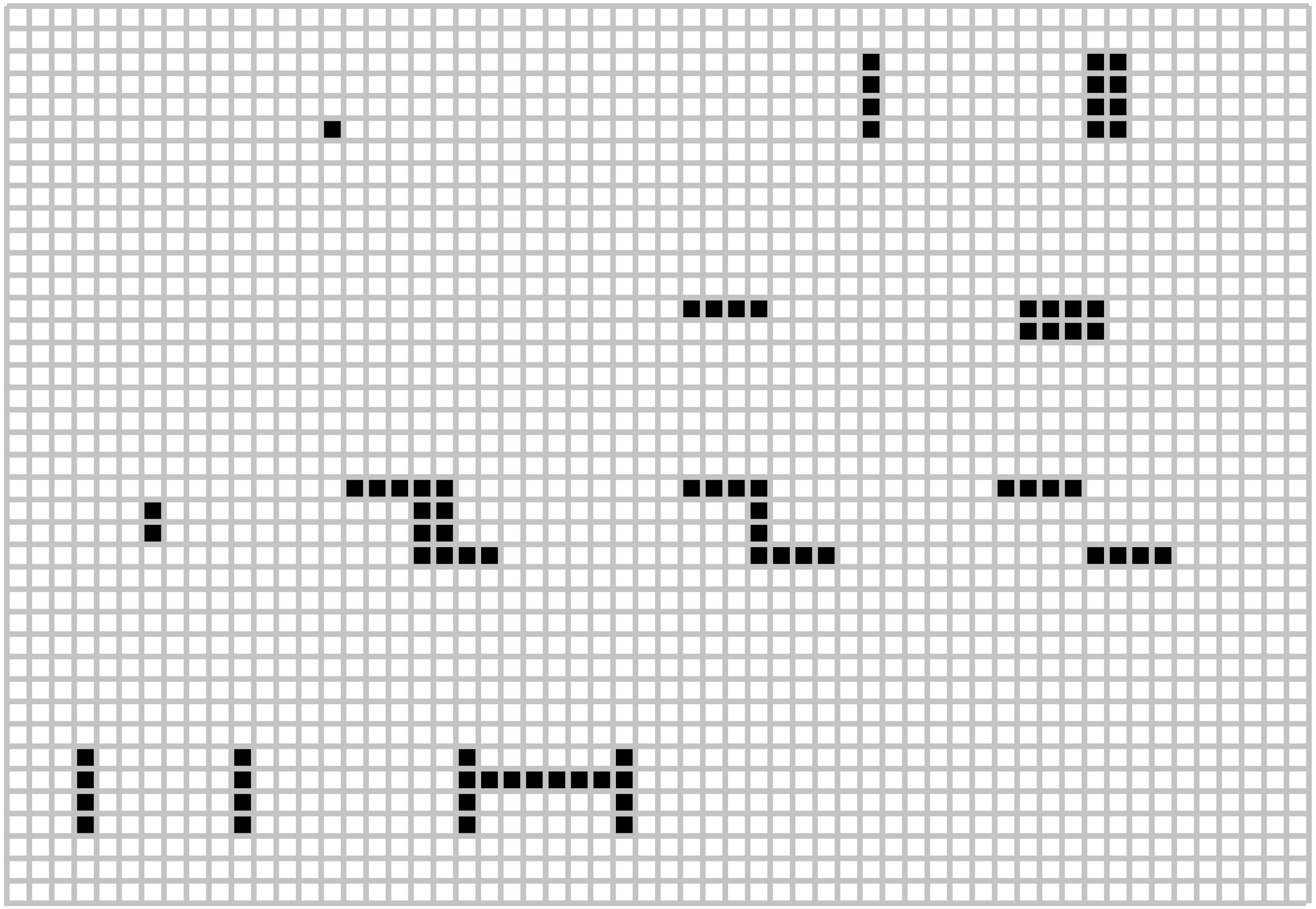
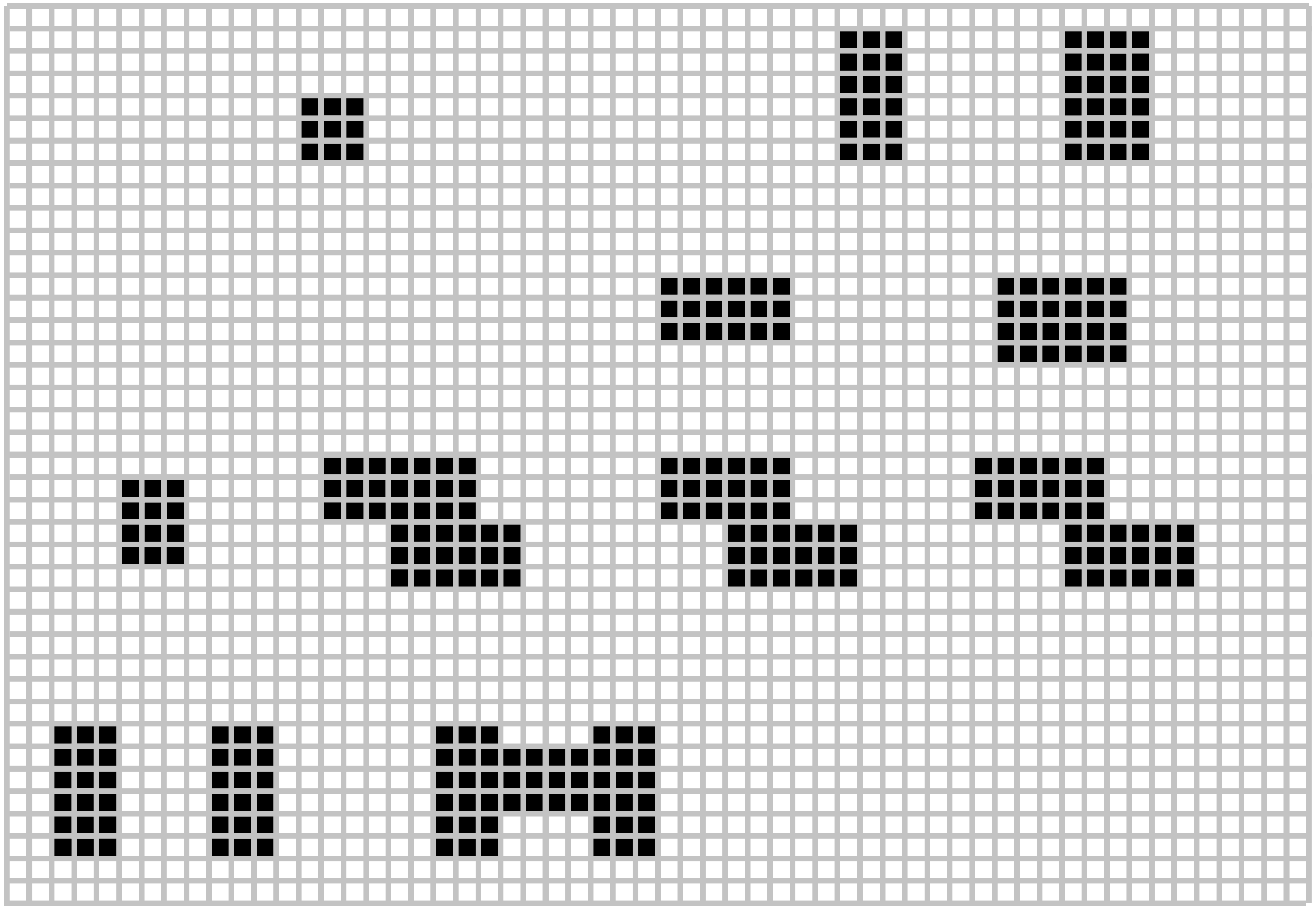
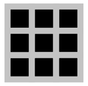


Image Operation – Open

Open: $X_S = (X \ominus S) \oplus S$ **– set of points in X touched by S as S is translated inside X.**



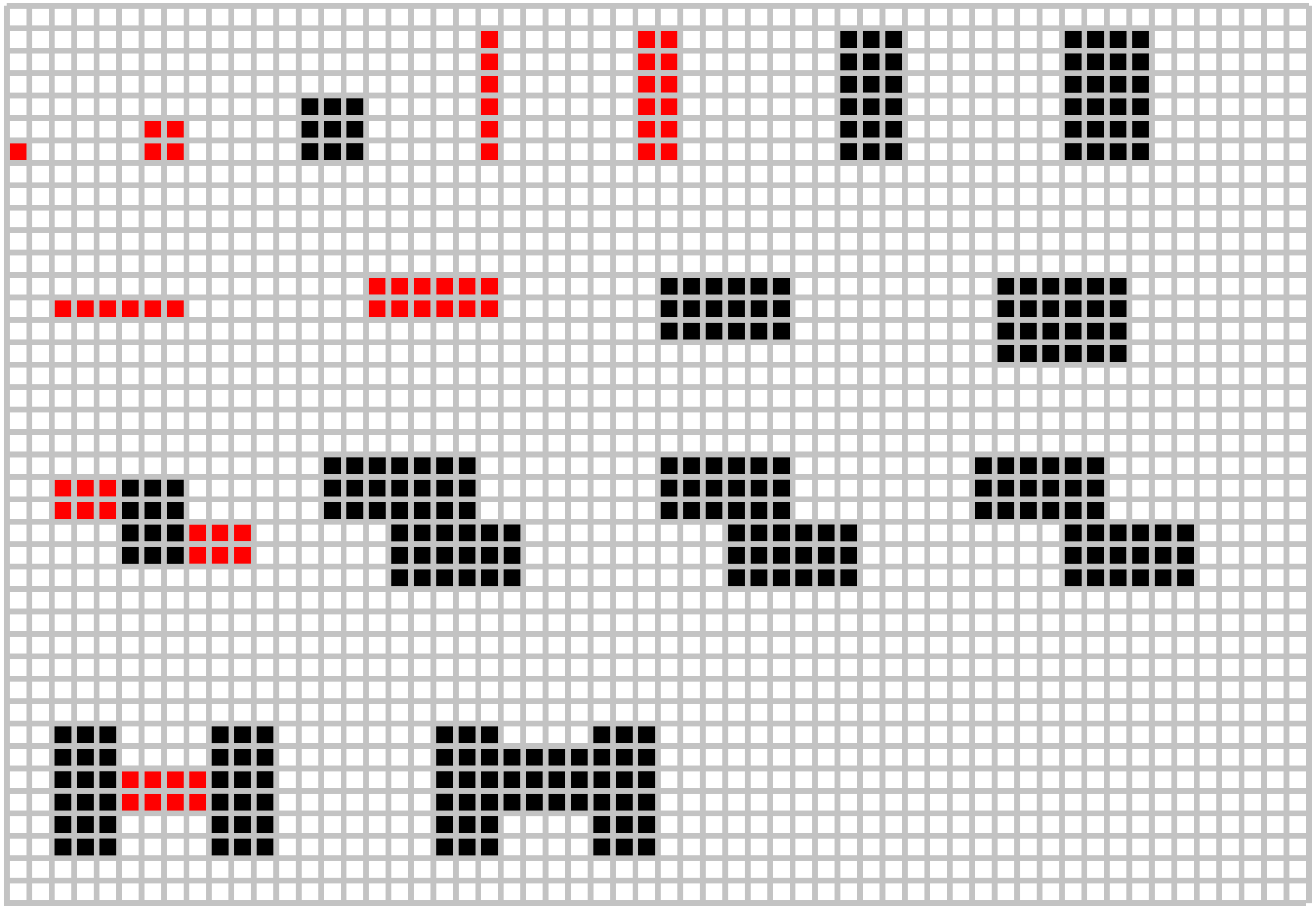
3λ Tolerance Check

$S(3) = Q(3) =$ **3×3 square** 

1. **Open Mask with $S(3)$**
   $\rightarrow$ anything not in open is an error
2. **$C \leftarrow$ Mask $\ominus S(3)$**
3. **Tag NE corner of each component of C**
   (*N.B.* tag is placed to NE of corner element)
4. **Dilate Tags (previous step) by $Q(3) \rightarrow$ any intersections with SW corners of different regions are errors**
5. **Tag NW corner of each component of C**
   (*N.B.* tag is placed to NW of corner element)
6. **Dilate (previous step) by $Q(3) \rightarrow$ any intersections with SE corners of different regions are errors**

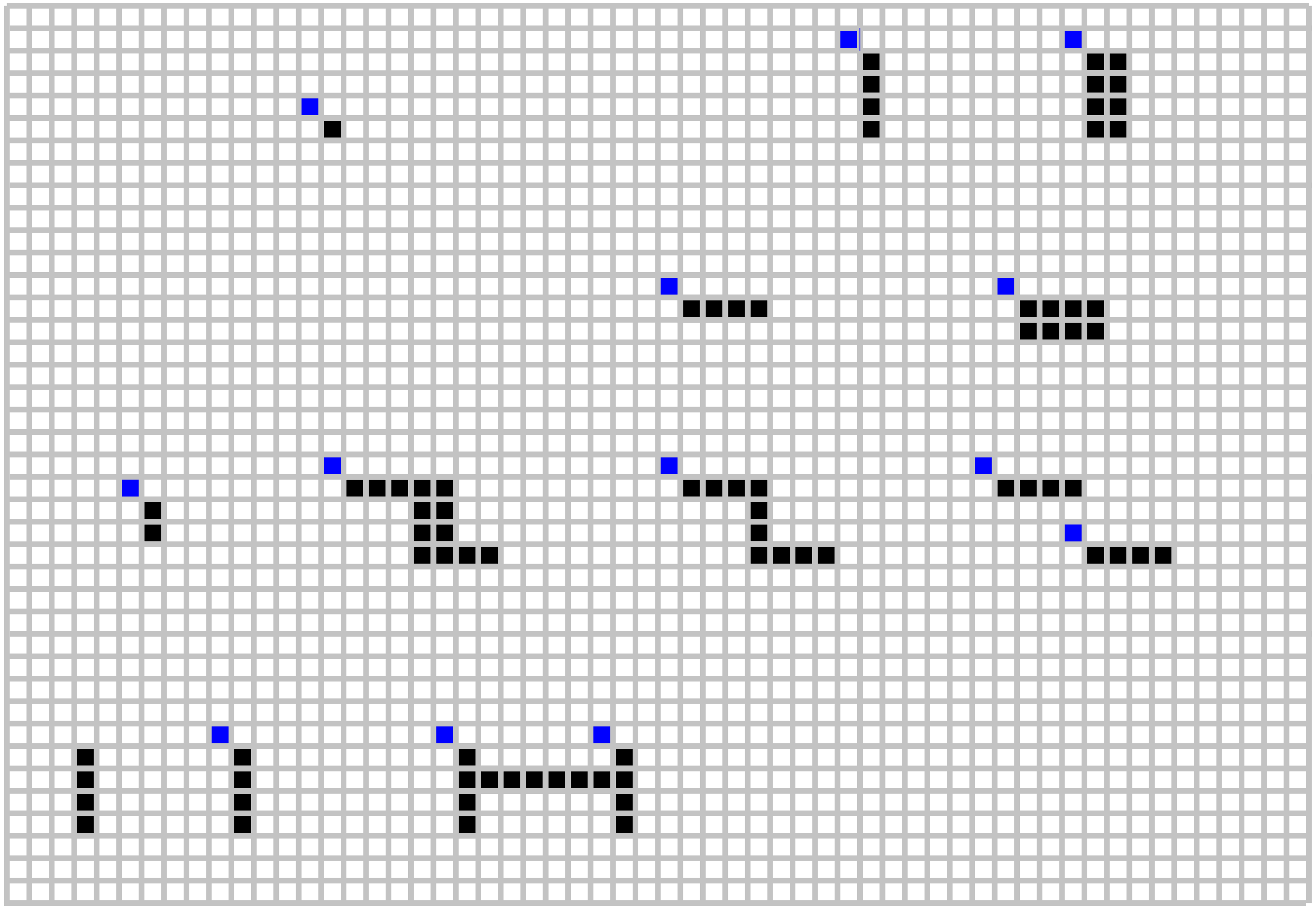
3λ Tolerance Check

After step 1: (errors in red – doesn't find diagonal error)



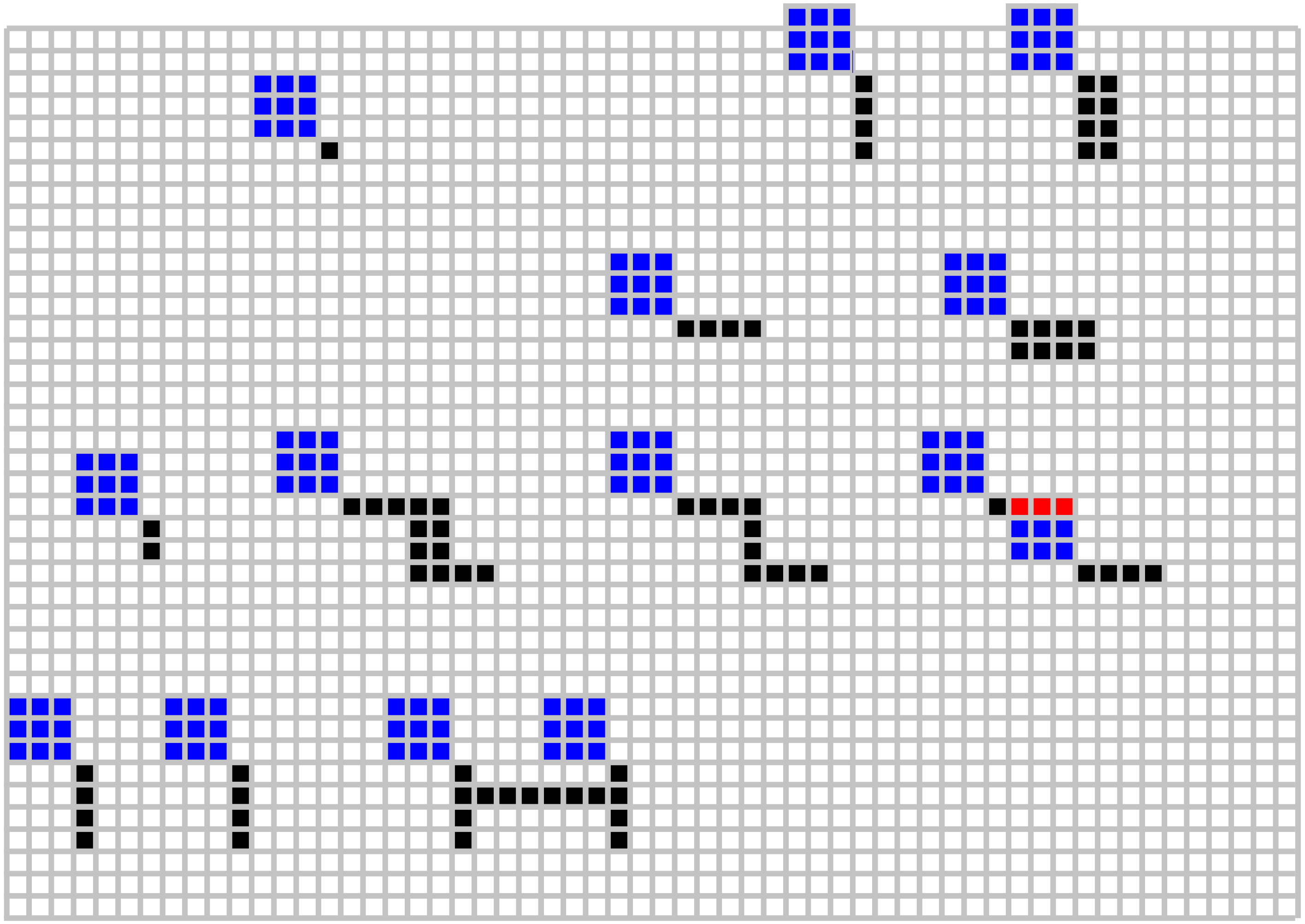
3λ Tolerance Check

After step 5: (*n.b.* tag (blue) to NW of corner point)



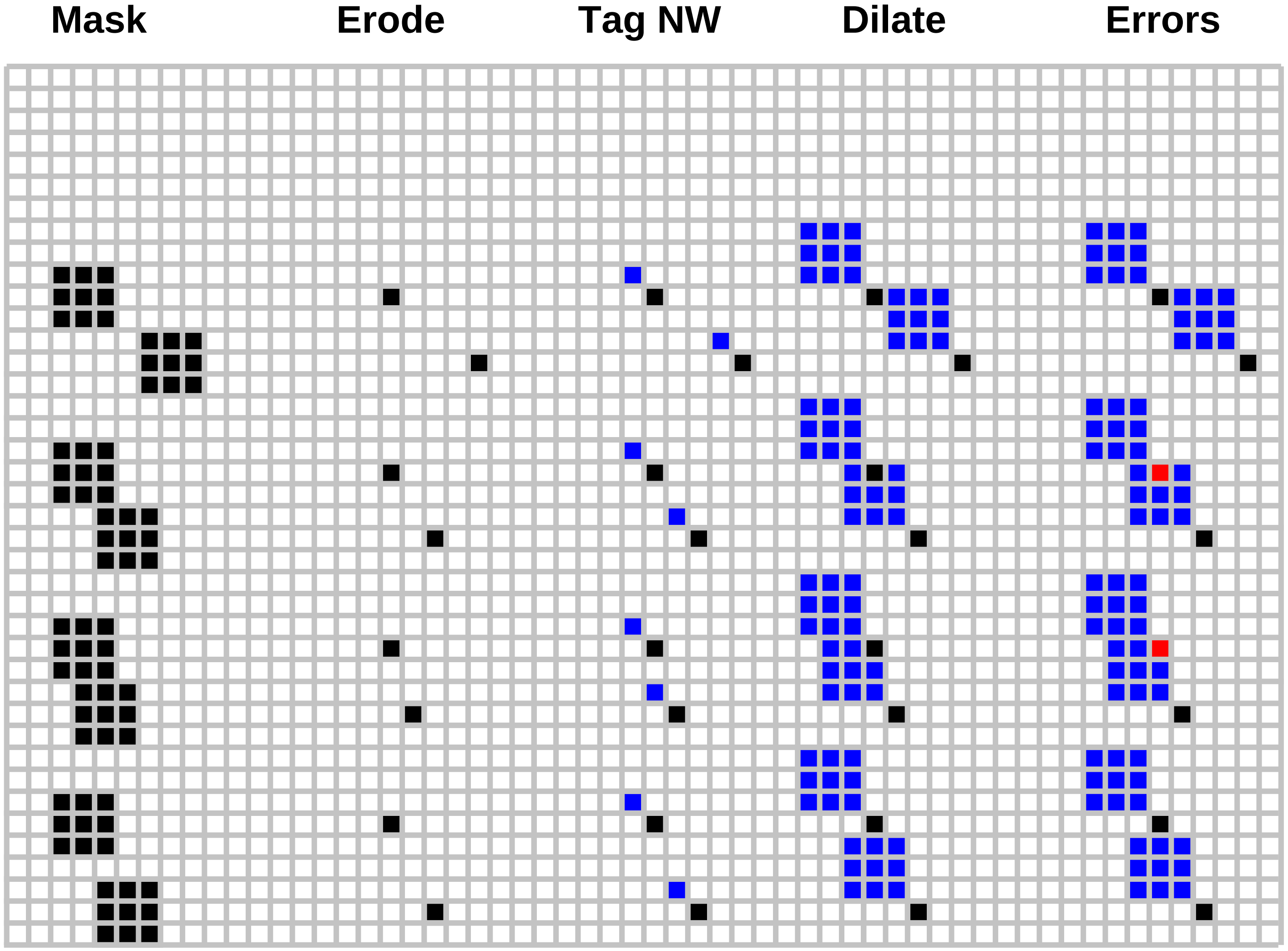
3λ Tolerance Check

After step 6: (errors in red – **finds** diagonal error)



3λ Tolerance Check

Check that this interpretation makes sense:



Summary

- Cellular Automata – simple FSMs, interacting in local neighborhood
- Useful for
  - modeling physical interactions
  - computing local, spatial relationships
- CA often >> hardware $\rightarrow$ partition
  in general, not have the directional flow which allowed convenient slicing in systolic arrays
  - raster pipeline
  - multigrid approach for some applications