

# Message-Passing Programming

---

## Cellular Automaton Exercise



# Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

[http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en\\_US](http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US)

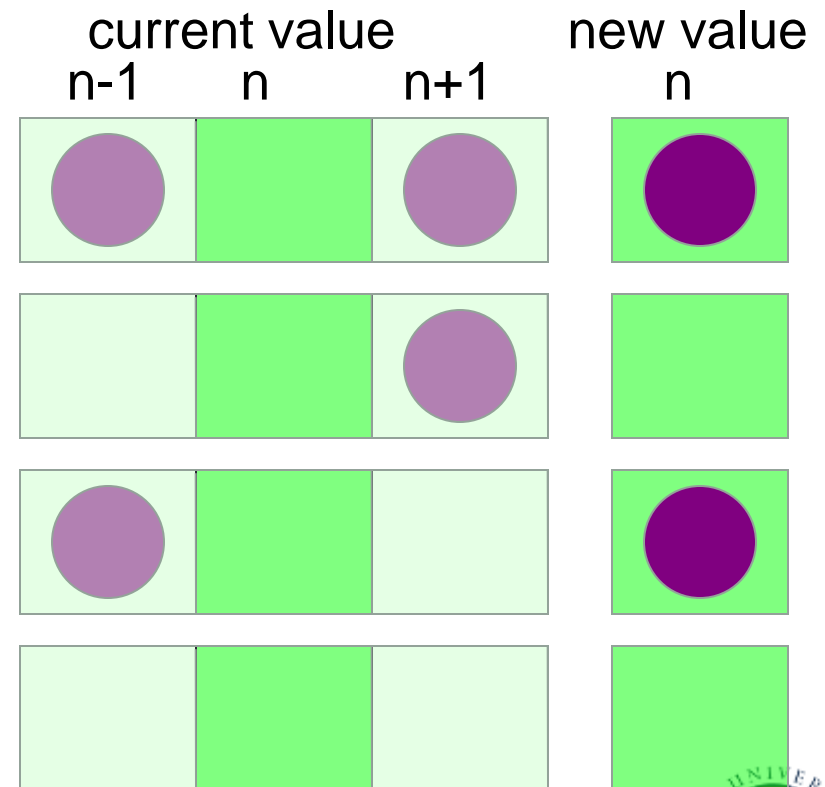
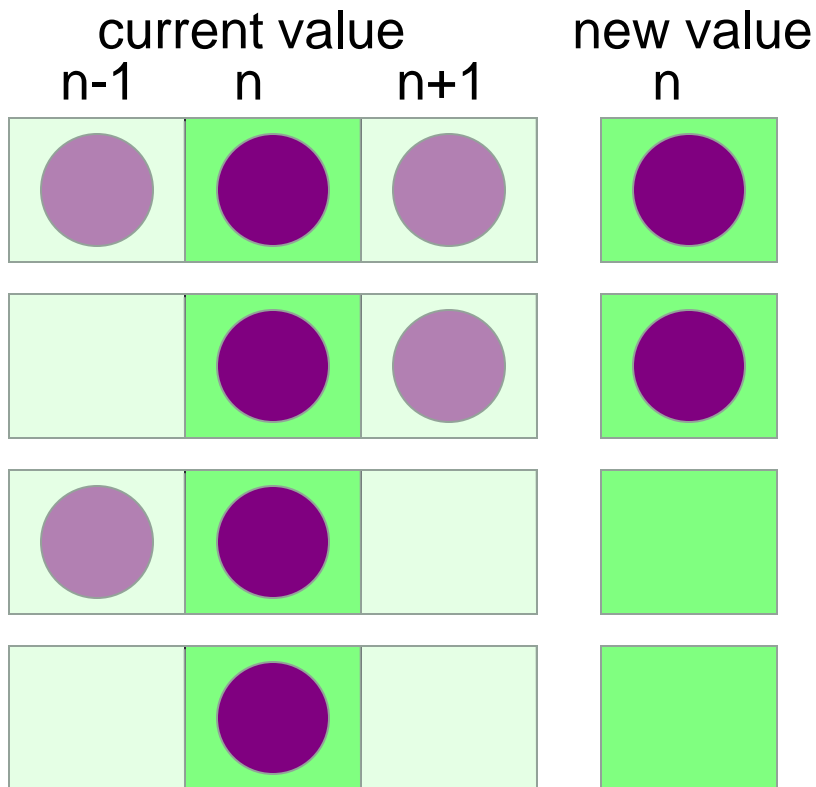
This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Acknowledge EPCC as follows: “© EPCC, The University of Edinburgh, [www.epcc.ed.ac.uk](http://www.epcc.ed.ac.uk)”

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

# Traffic simulation

- Update rules depend on:
  - state of cell
  - state of nearest neighbours in both directions



# State Table

- If  $R^t(i) = 0$ , then  $R^{t+1}(i)$  is given by:

	$R^t(i-1) = 0$	$R^t(i-1) = 1$
$R^t(i+1) = 0$	0	1
$R^t(i+1) = 1$	0	1

- If  $R^t(i) = 1$ , then  $R^{t+1}(i)$  is given by:

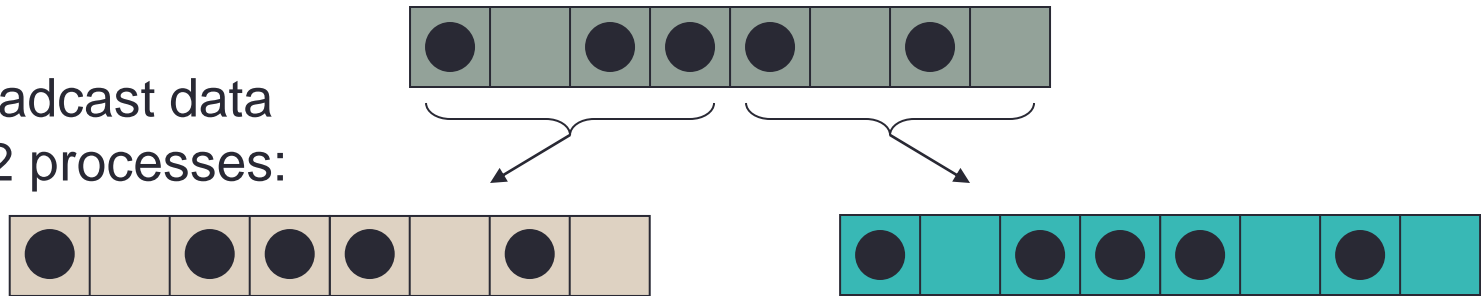
	$R^t(i-1) = 0$	$R^t(i-1) = 1$
$R^t(i+1) = 0$	0	0
$R^t(i+1) = 1$	1	1

# Pseudo Code

```
declare arrays old(i) and new(i), i = 0,1,...,N,N+1
initialise old(i) for i = 1,2,...,N-1,N (eg randomly)
loop over iterations
    set old(0) = old(N) and set old(N+1) = old(1)
    loop over i = 1,...,N
        if old(i) = 1
            if old(i+1) = 1 then new(i) = 1 else new(i) = 0
        if old(i) = 0
            if old(i-1) = 1 then new(i) = 1 else new(i) = 0
    end loop over i
    set old(i) = new(i) for i = 1,2,...,N-1,N
end loop over iterations
```

# Parallelisation Strategy (1)

Broadcast data  
to 2 processes:



Split calculation  
between 2 processes:



Process 1

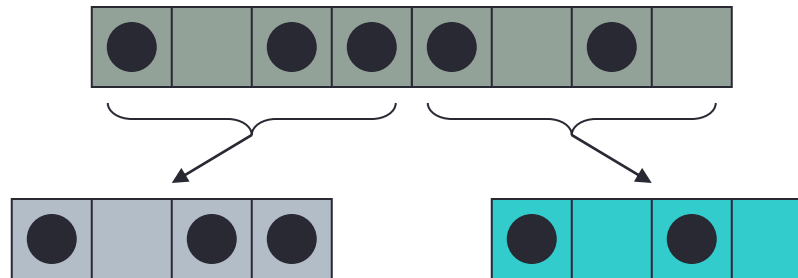


Process 2

- Globally resynchronise all data after each move
  - a **replicated data** strategy
- Every process stores the entire state of the calculation
  - e.g. any process can compute total number of moves

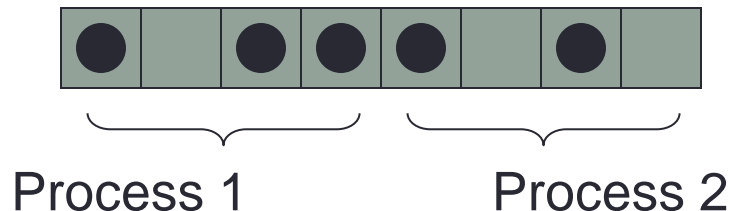
# Parallelisation Strategy (2)

Scatter data  
between 2 processes:  
**distributed data strategy**



- Internal cells can be updated independently.
- Must communicate with neighbouring processes to update edge cells.
- Sum local number of moves on each process to obtain total number of moves at each iteration.

Split calculation  
between 2 processes:



- Each process must know which part of roadway it is updating.
- Synchronise at completion of each iteration and obtain total number of moves.

# Parallelisation

- Load balance not an issue
  - updates take equal computation regardless of state of road
  - split the road into equal pieces of size  $N/P$
- For each piece
  - rule for cell  $i$  depends on cells  $i-1$  and  $i+1$
  - the  $N/P - 2$  interior cells can be updated independently in parallel
  - however, the edge cells are updated by other processors
    - similar to having separate rules for boundary conditions
- Communications required
  - to get value of edge cells from other processors
  - to produce a global sum of the number of cars that move



# Message Passing Parallelisation



2 processes,  
add halos



copy data  
to halos



update  
interior cells



local moves = 1

local moves = 2

global moves = 3