HIC 2012

CADDIES: a New Framework for Rapid Development of Parallel Cellular Automata Algorithms for Flood Simulation

M. Guidolin, A. Duncan, B. Ghimire, M. Gibson, E. C. Keedwell, A. S. Chen, S. Djordjević and D. A. Savić

Centre for Water Systems, College of Engineering, Mathematics and Physical Sciences , University of Exeter, Exeter, UK

UNIVERSITY OF EXETER

Centre for Water Systems

Project Partners & Team

City of Bradford MDC www.bradford.gov.uk
Environment Agency
Halcrow
HR Wallingford Working with water
mouchel
MWH BUILDING A BETTER
UKWIR
United Utilities
NORTHUMBRIAN WATER
raa Richard Allitt Associates Ltd
SOUTH WEST WATER
Thames Water
TORBAY COUNCIL
MWH Soft
YorkshireWater


Prof Dragan Savic


Prof Slobodan Djordjevic


Dr Edward Keedwell


Dr Albert Chen


Dr Bidur Ghimire


Dr Michele Guidolin


Miss Rebecca Austin


Mr Mike Gibson


Mr Andrew Duncan

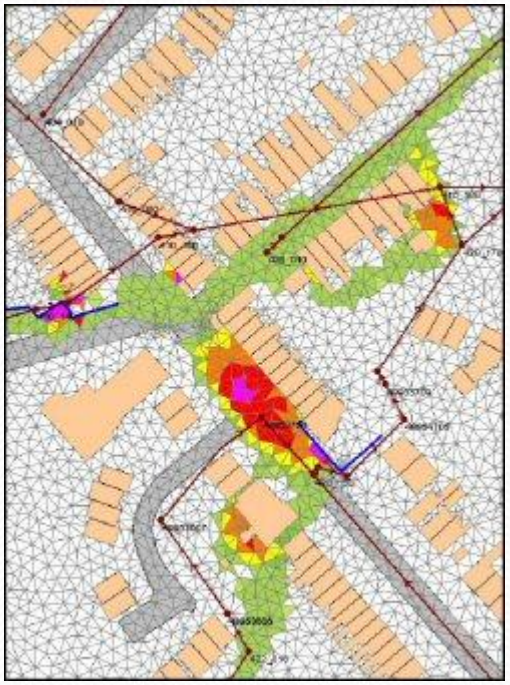
UNIVERSITY OF EXETER
Centre for Water Systems

Outline

- Conventional pluvial flood modelling
- CADDIES project
  - Cellular Automata Dual DraInagE Simulation
  - Cellular Automata (CA) modelling alternative
- CA Framework (API)
  - Research tool to allow CA models to be developed
  - Parallelisation (GPU / Cluster / Cloud)
  - CA model performance evaluation

Conventional 1D/2D Pluvial Flood Modelling

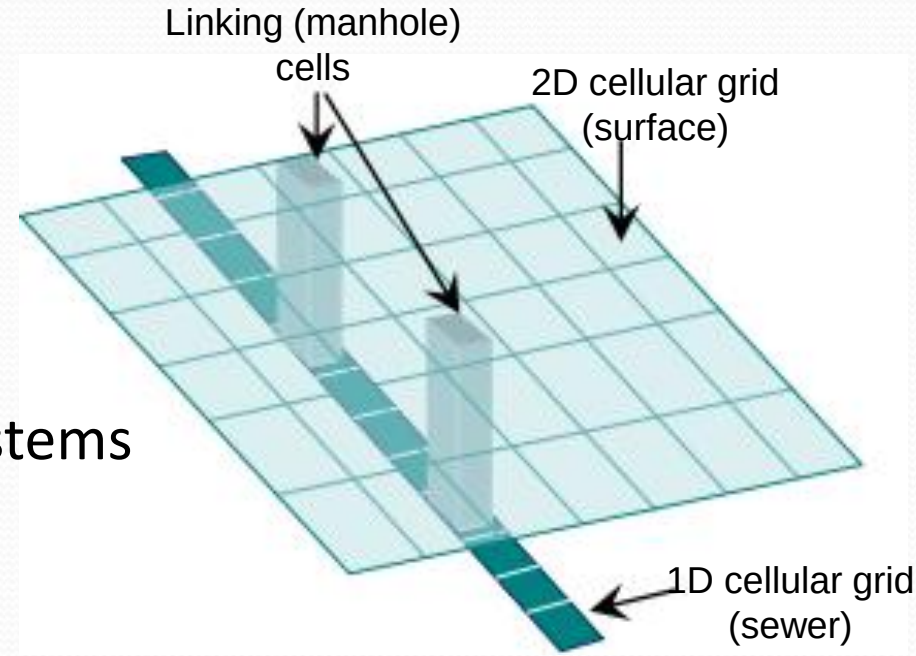
- Hydrodynamic (St Venant equations)
  - Solution is iterative process
  - Urban surface / sewer flow
    - "dual drainage"
  - Full 2D approach
    - computationally intensive (hours run time)
  - Pseudo-2D
    - 1D surface channels + ponds
- Trade-off between speed and accuracy



CADDIES project

Cellular Automata Dual DraInagE Simulation

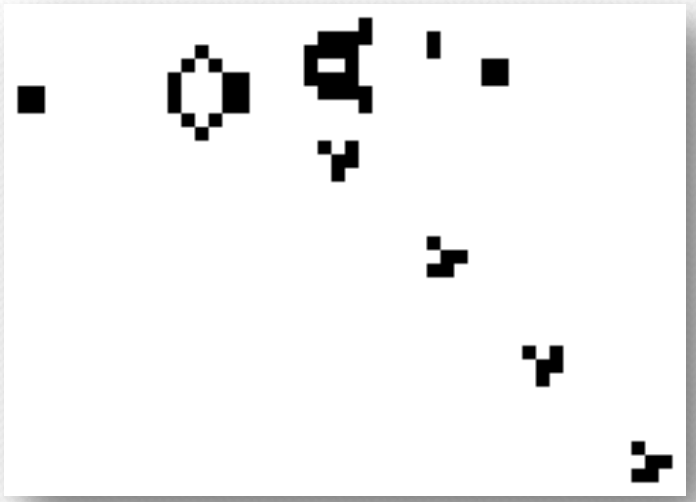
- Improve “Dual-Drainage” flood modelling
  - Urban surface / sewer flow
- Cellular Automata
  - Simulate complex physical systems using simple rules
  - Fast computations



UNIVERSITY OF EXETER
Centre for Water Systems

Cellular automata

- Grid of cells
- Each cell has a set of states
- Simple transition rules on each cell
- Use previous state of neighbourhood
- Ideal for parallelisation
- Famous example
  - Game of life (glider gun)

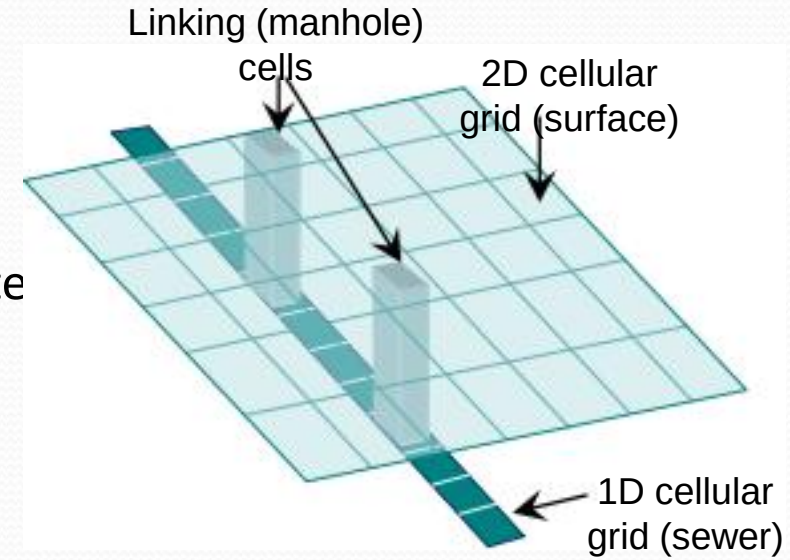


CADDIES Challenges

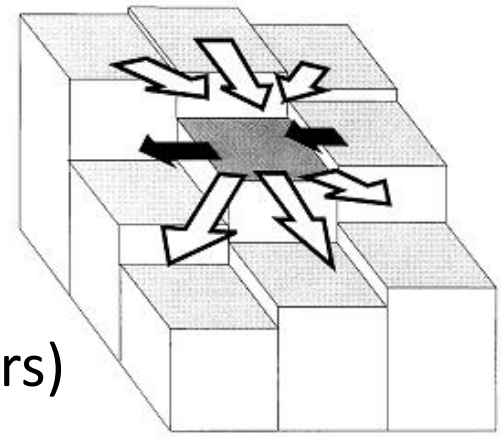
To simulate complex flooding processes

using simple rules... not trivial!

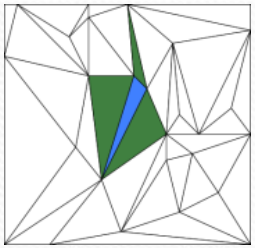
- Grid of cells – each with a "state"
  - e.g. flood depth / flow rate / water volume
- Cell states evolve stepwise
  - simultaneously across whole grid
- New state of each cell
  - only determined by states of immediate neighbours in previous timestep

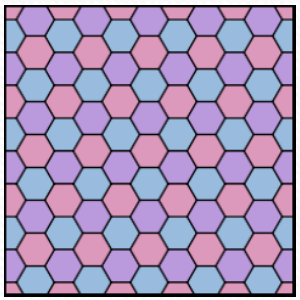


CADDIES Challenges



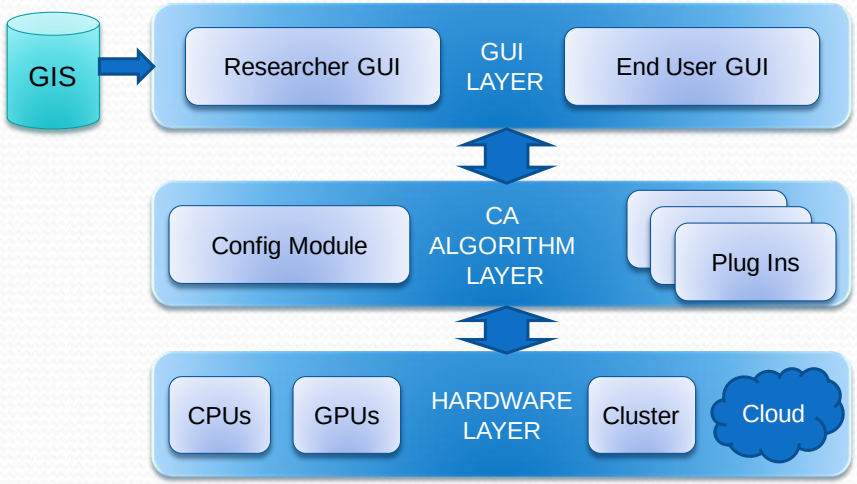
- CA Algorithm is defined by:
  - Transition rules (based on states of neighbours)
    - How many states and which types of value?
    - Which relationships / transition functions?
    - How to handle grid boundary cells?
  - Underlying grid structure
    - Which types of grid structure? (regular / irregular)
    - Cell shapes? (polygon sides)
    - How many cells in the neighbourhood?





CADDIES Framework

- A new flexible software environment (various tools)
  - API (Application Programming Interface)
  - that aims to:
    - Simplify the development/test/analysis of CA algorithms
    - Accelerate CA algorithm using modern hardware
    - Graphically manage the development of CA algorithms



CADDIES CA API

- Standard set of
  - Data structures
  - Variables
  - Methods

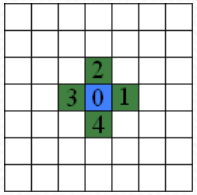
```
1  CA_FUNCTION outflow(CA_GRID grid,
2                      CA_EDGEBUFF_REAL_IO OUTF,
3                      CA_CELLBUFF_REAL_I  ELV,
4                      CA_EDGEBUFF_REAL_I  WL) {
5    CA_GRID_INIT(grid);
6    CA_ARRAY_CREATE(grid, CA_REAL, aelv,  caNeighbours+1);
7    CA_ARRAY_CREATE(grid, CA_REAL, aoutf, caEdges+1);
8    CA_INDEX index = caIndex(grid,0);
9    ...
10   caReadCellBuffRealCellArray(grid, ELV, aelv);
11   caReadEdgeBuffRealEdgeArray(grid, ELV, index, aoutf);
```

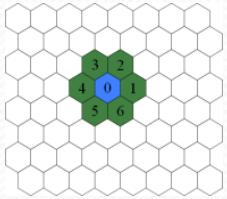
```
1  #include CA_2D_INCLUDE(outflow)
2
3  int main(...){
4   ...
5   CA::Grid          GRID(ncols,nrows,cellsize);
6   CA::CellBuffReal  ELV(GRID);          // Elevation
7   CA::CellBuffReal  WL(GRID);           // Water Level
8   CA::EdgeBuffReal  OUTF(GRID);         // Outflow
9   CA::Box           area(GRID.box());   // Computational Area
10  ...
11  while(loop) {
12   // Execution of the "outflow" CA transition function
13  CA::Execute::function(area,outflow,GRID,OUTF,ELV,WL);
14   ...
15  }}
```

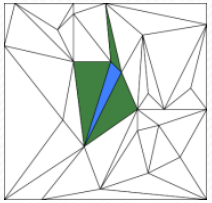
UNIVERSITY OF EXETER
Centre for Water Systems

CADDIES CA API Ideas

- The developer need write the CA algorithm only once
- The algorithm will work with different types of grids





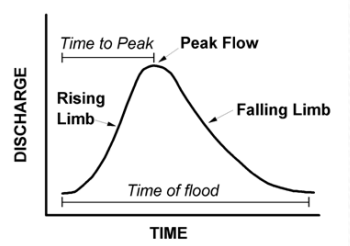


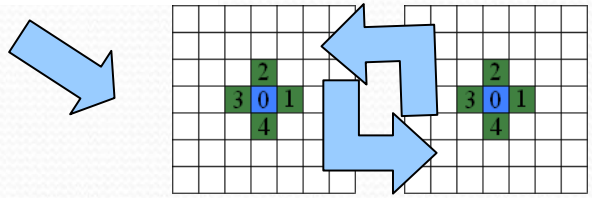
- Automatically accelerate execution using modern hardware (GPU / Cluster / Cloud)

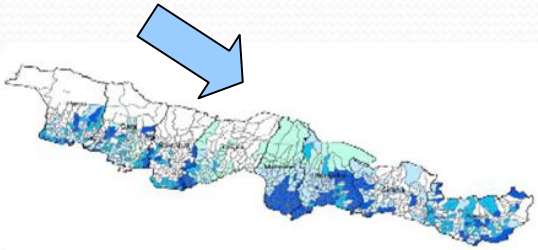


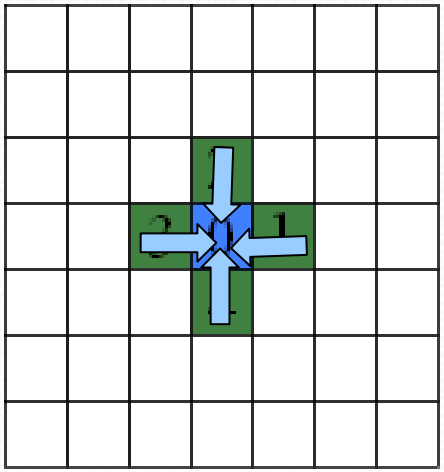
CA ALGORITHM using the API

1. CA Execution
   - Data management
   - Flow control
2. CA Transition Rules
   - Computation in each cell



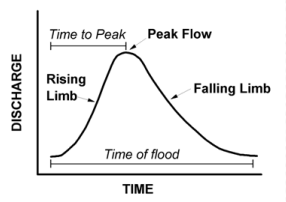


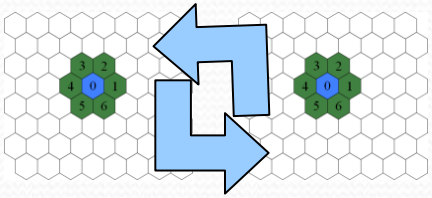


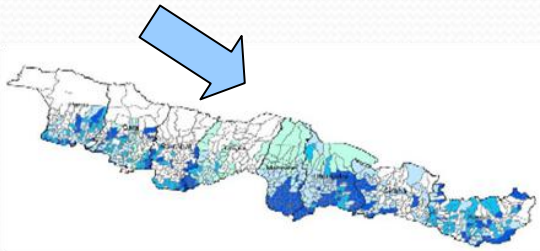


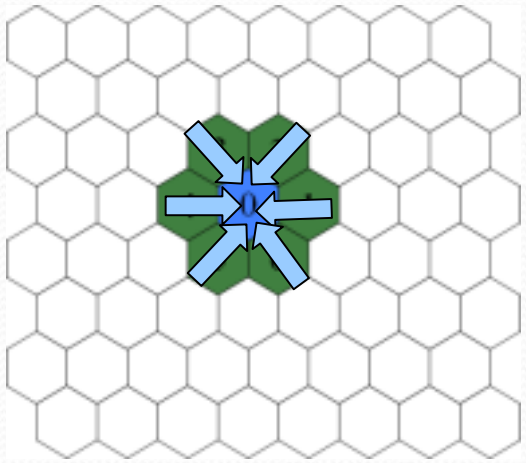
CA ALGORITHM using the API

3. CA options
   - Define Attributes
     - Grid/Cells/Etc.







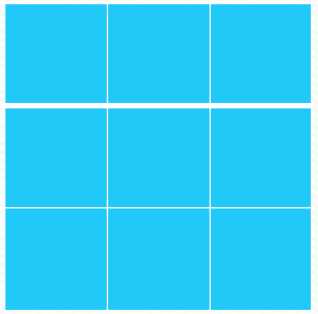


CA API Implementations

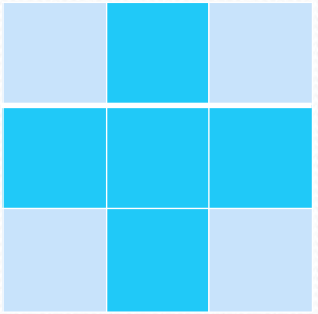
- Different implementations depending on:
  - CA attributes
  - Hardware used
- At the moment three implementations:
  - Sequential CPU / Square / Moore
  - Sequential CPU / Square / von Neumann
  - OpenCL GPU / Square / von Neumann

Neighbourhoods

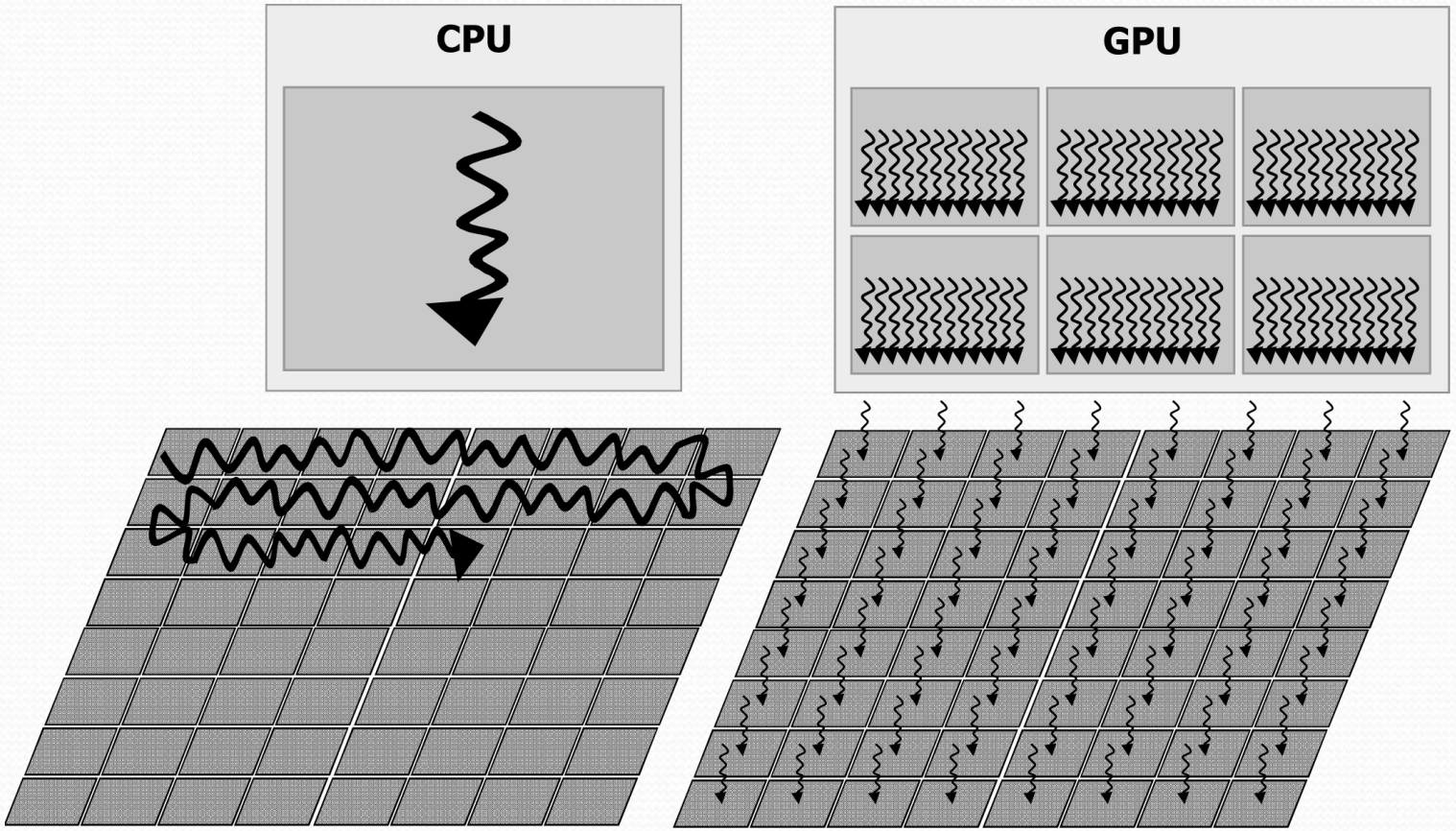
Moore



von Neumann



CPU versus GPU



Game of Life Example

CA Execution

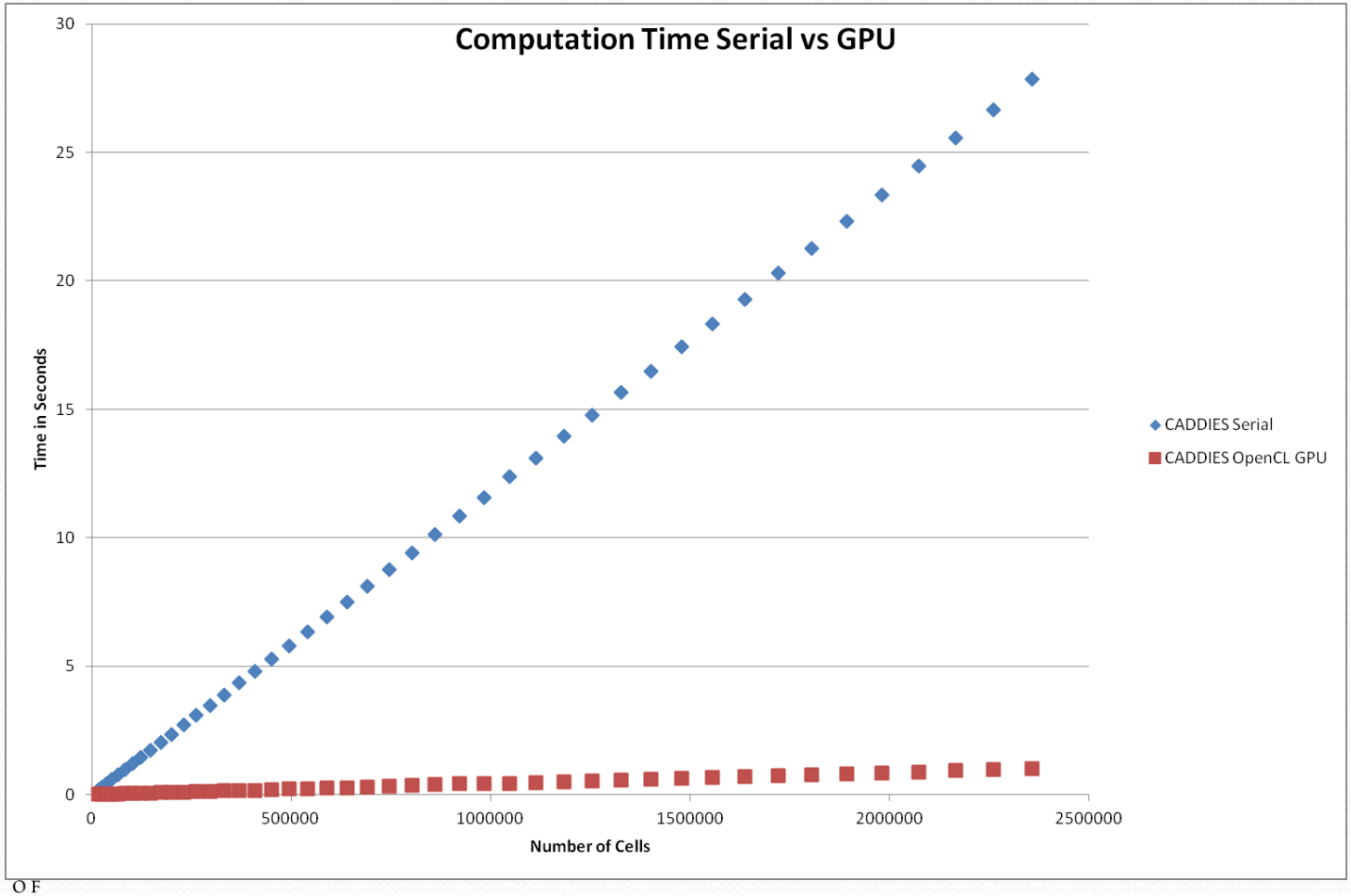
```
#include CA_2d_INCLUDE(gol)

int main(...){
  ...
  CA::Grid          GRID(ncols,nrows,1);
  CA::CellBuffState A(GRID),B(GRID);
  CA::Box           area(GRID.box());
  ...
  while(loop) {
    CA::Execute::function(area,gol,GRID,A,B);
    ...
  }
}
```

CA Transition Rules

```
CA_FUNCTION gol(CA_GRID grid,
                CA_CELLBUFF_STATE_IO a,
                CA_CELLBUFF_STATE_I  b)
{
  CA_GRID_INIT(grid);
  CA_ARRAY_CREATE(grid,CA_STATE,states,caNeighbours+1);
  ...
  caReadCellBuffStateCellArray(grid,b,states);
  for(int i=0;i<=caNeighbours;i++) {
     ...
  }
  caWriteCellBuffState(grid,a,0,states[0]);
}
```

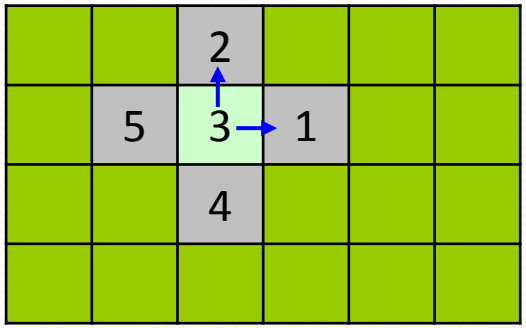
UNIVERSITY OF EXETER
Centre for Water Systems

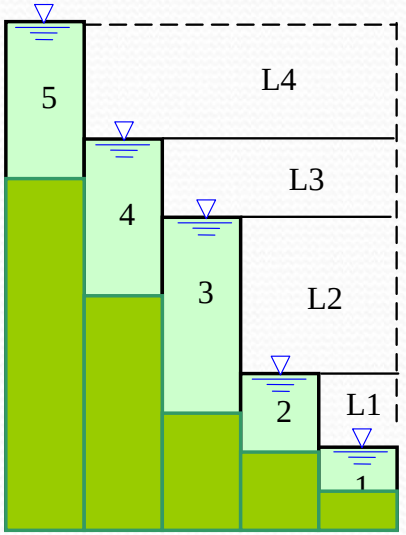
Game of Life Results



CA Flood algorithm Example

- Computes outflow of cell by ranking the water surface elevation





- Presented in detail in HIC 2012 paper:
  - Colleague: Dr. Bidur Ghimire
  - Wednesday 18 July 9.20 - 9.40
  - Topic A5.2

Case Study

- Stockbridge area in Keighley (UK)
- DEM of 2m resolution with 377x269 cells
- Rainfall of 42.3 mm/hr (100 year return)
- Used two CA API implementations
  - Sequential CPU / Square / von Neumann
  - OpenCL GPU / Square / von Neumann
- Used two hardware configurations
- Used physical based UIM model as reference

CA flood algorithm Results

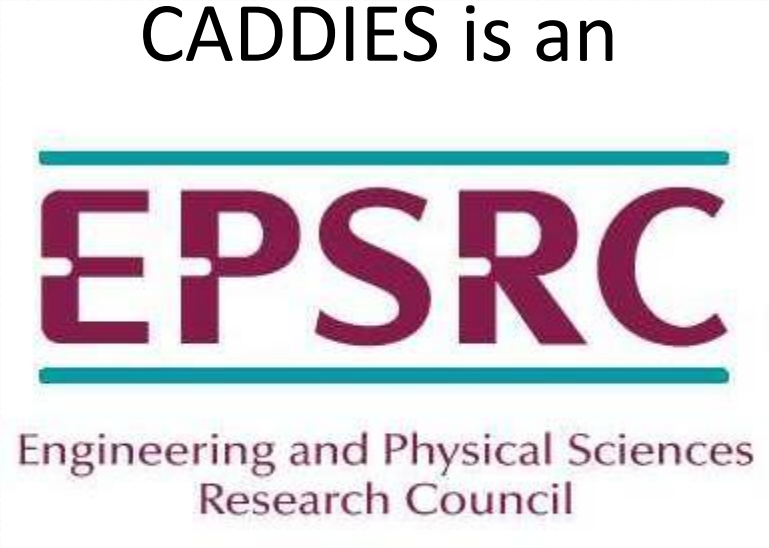
	UIM	Simple Serial			OpenCL			
Model	Time (s)	Time (s)	S_p^{UIM}	RMSE (m)	Time (s)	S_p^{Ser}	S_p^{UIM}	RMSE (m)
1	10371.1	280.9	36.9	0.027	50.8	5.5	204.2	0.027
2	5885.7	161.6	36.4	0.029	44.5	3.6	132.3	0.029

Conclusion

- CA API is part of CADDIES framework
- It possible to simply develop CA algorithms
  - Game of life
  - CA algorithms for pluvial flood modelling
- Automatically accelerate algorithm using modern hardware
- CA flood model with GPU can be an order of magnitude faster than physical models (UIM)

Acknowledgment

CADDIES is an



Funded Project (Number: GR/J09796)

Thanks

M.Guidolin@exeter.ac.uk
http://centres.exeter.ac.uk/cws

UNIVERSITY OF EXETER
Centre for Water Systems