Shallow Water Waves on a Deep Technology Stack: Accelerating a Finite Volume Tsunami Model using Reconfigurable Hardware in Invasive Computing

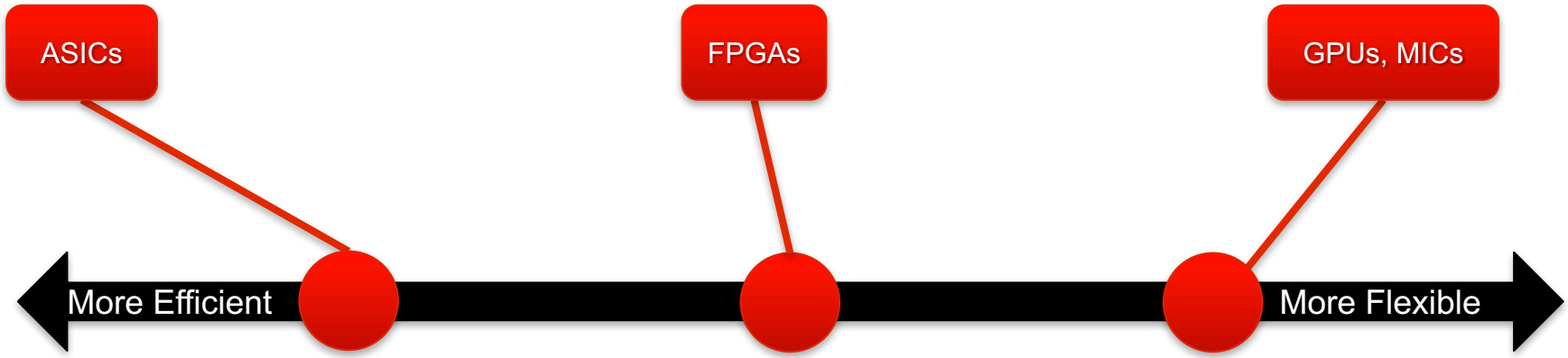
Alexander Pöppl[1], Marvin Damschen[2], Florian Schmaus[3], Andreas Fried[2], Manuel Mohr[2], Matthias Blankertz[2], Lars Bauer[2], Jörg Henkel[2], Wolfgang Schröder-Preikschat[3], Michael Bader[1]

[1]) Technical University of Munich, Germany
[2]) Karlsruhe Institute of Technology, Germany
[3]) Friedrich-Alexander University Erlangen-Nürnberg, Germany

The 10th Workshop on UnConventional High Performance Computing 2017
August 28th, Santiago de Compostela, Spain

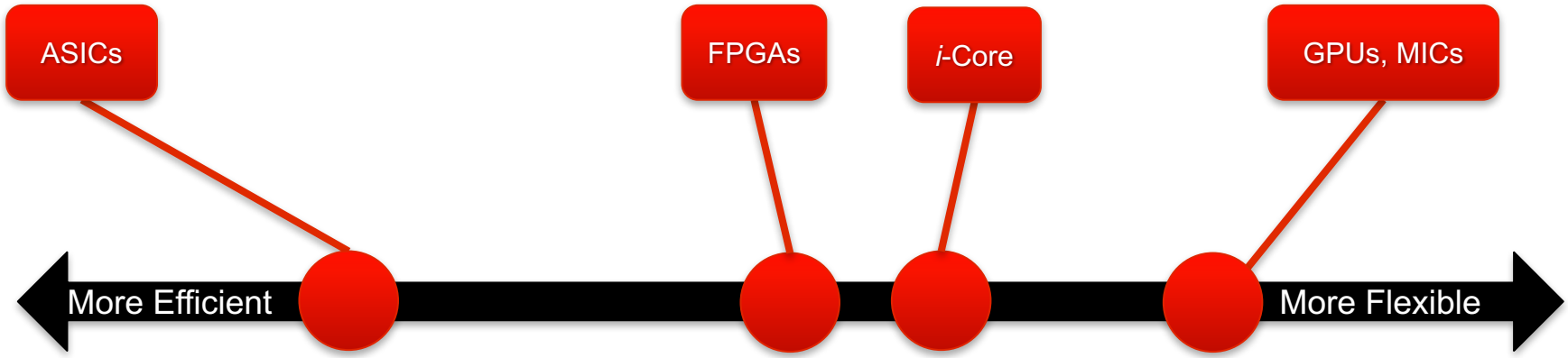
Motivation

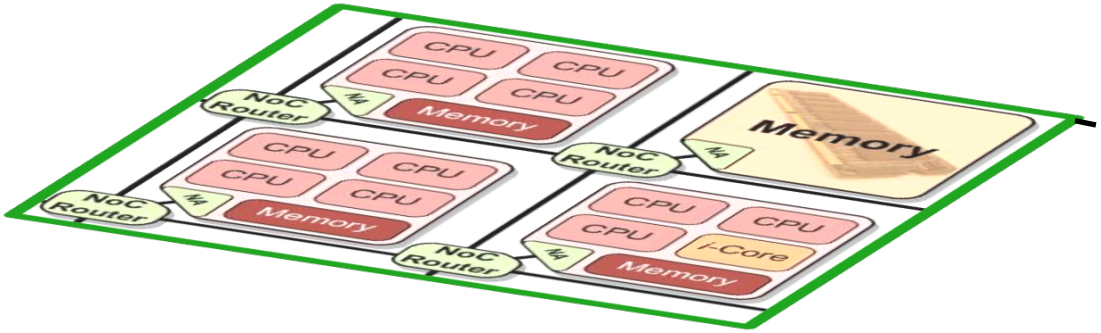
- Heterogeneous Environments commonplace in HPC
  - NVidia Tesla GPUs, Intel Xeon Phi, …
  - New: Application-specific hardware (Google Tensor Processing Units, Microsoft Catapult, Anton 2)
- Reconfigurable fabric commonly used in embedded scenarios
  - Performance comparable to ASICs
  - May be reconfigured at run time.
  - Special case: Reconfigurable fabric and CPU **on the same chip**



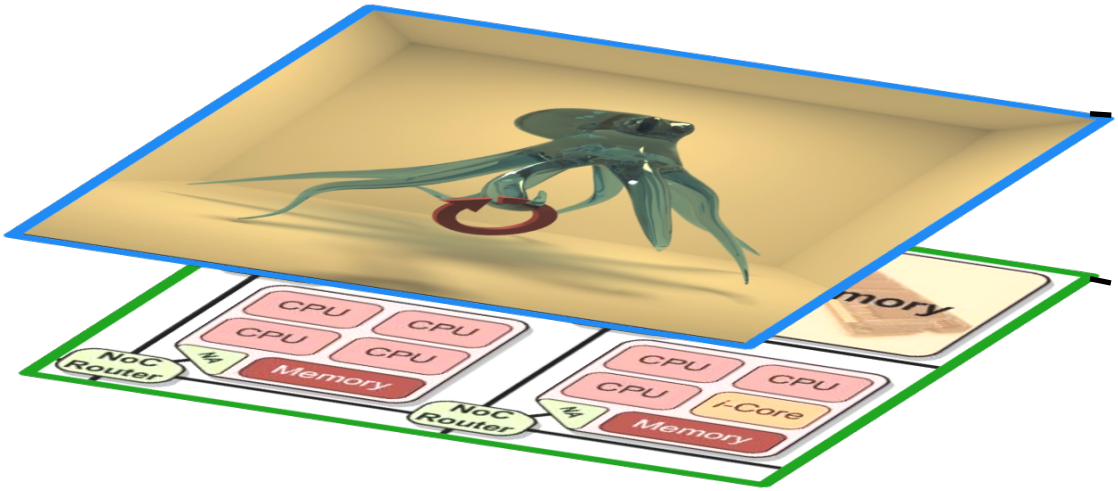
Motivation

- Heterogeneous Environments commonplace in HPC
  - NVidia Tesla GPUs, Intel Xeon Phi, …
  - New: Application-specific hardware (Google Tensor Processing Units, Microsoft Catapult, Anton 2)
- Reconfigurable fabric commonly used in embedded scenarios
  - Performance comparable to ASICs
  - May be reconfigured at run time.
  - Special case: Reconfigurable fabric and CPU **on the same chip**



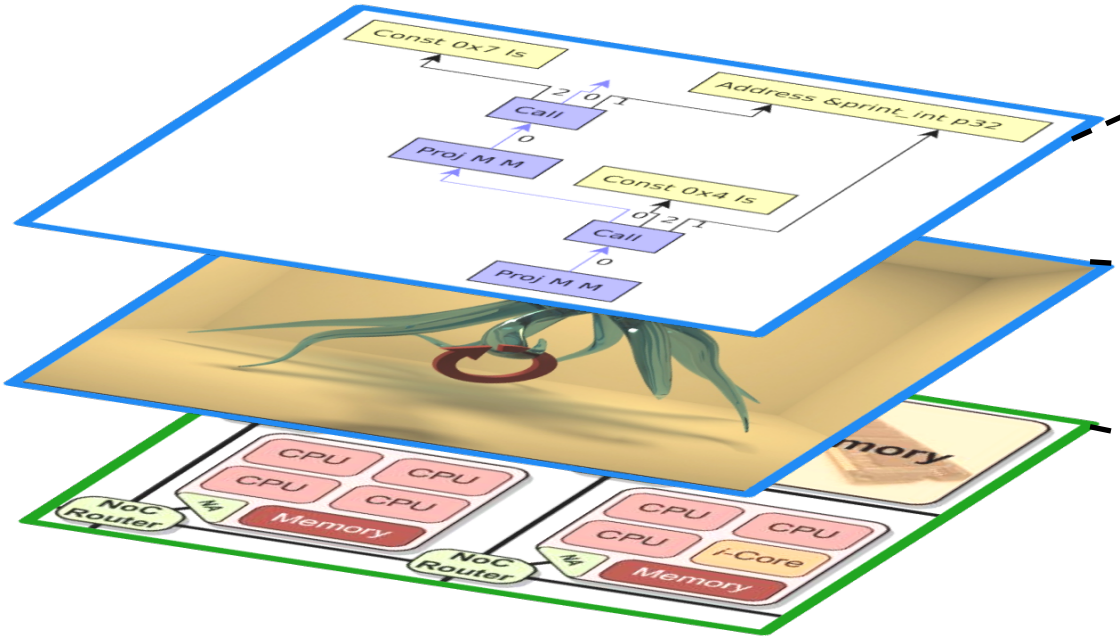


InvasIC Architecture
Heterogeneous Tiled Architecture



OctoPOS
Parallel Operating System

InvasIC Architecture
Heterogeneous Tiled Architecture

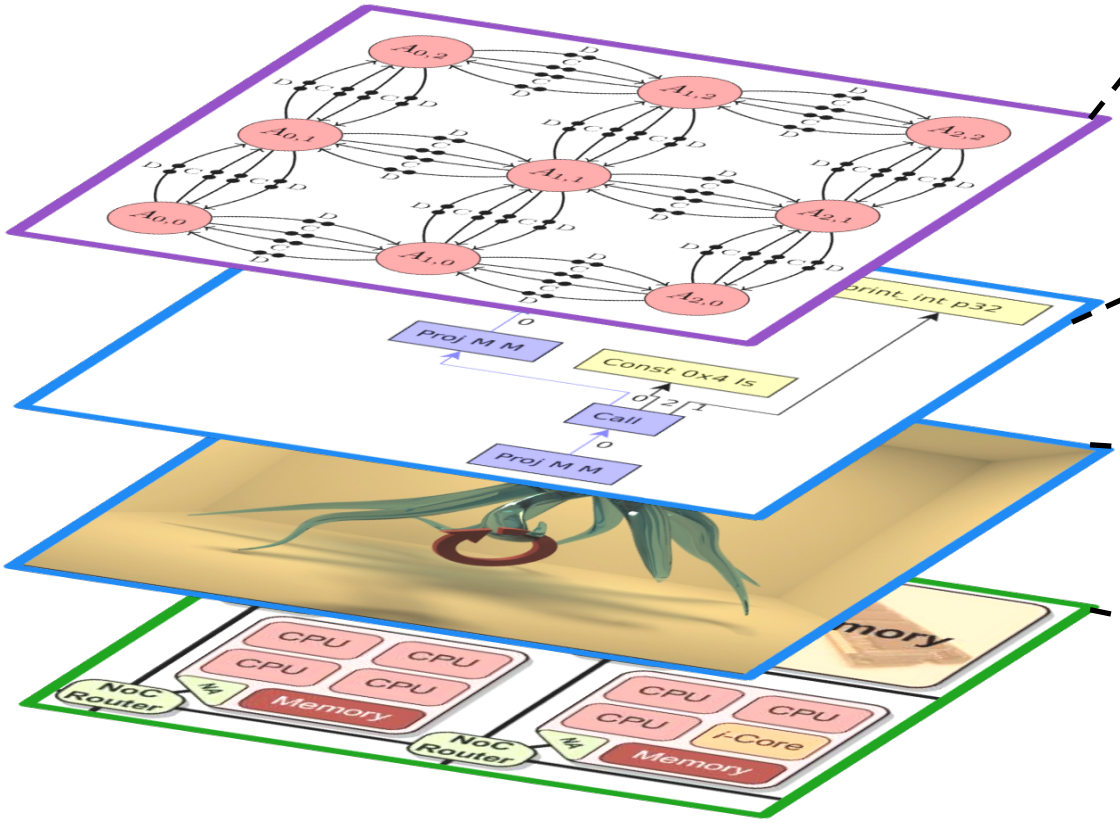


X10i (based on libFIRM)
Invasive X10 Compiler and Runtime

OctoPOS
Parallel Operating System

InvasIC Architecture
Heterogeneous Tiled Architecture

Invasive Computing



SWE-X10 (in ActorX10)
Application utilizing *i*-Core Acceleration

X10i (based on libFIRM)
Invasive X10 Compiler and Runtime

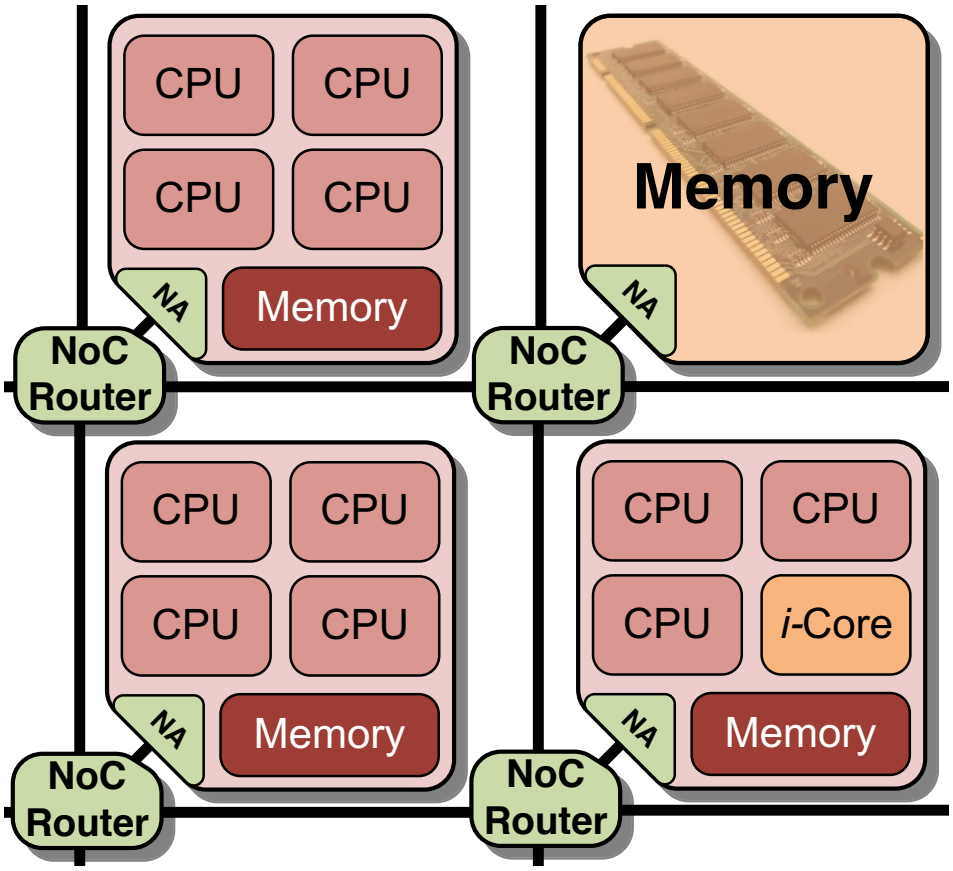
OctoPOS
Parallel Operating System

InvasIC Architecture
Heterogeneous Tiled Architecture

Application | Middleware | Hardware

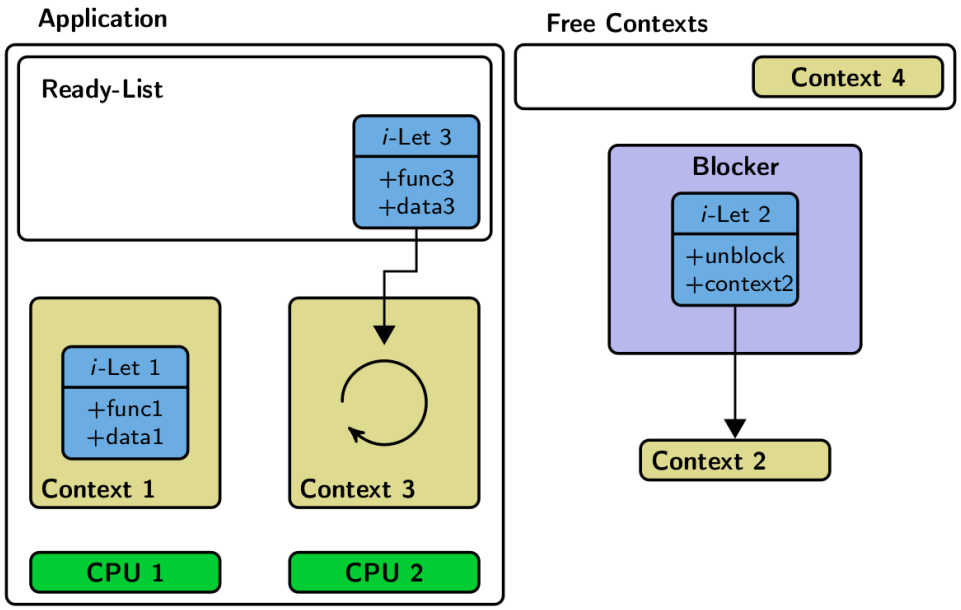
The InvasIC Hardware Architecture

- Heterogeneous Multiprocessor System-on-Chip
- Tiled Architecture
  - RISC Tiles
  - *i*-Core Tiles
  - Memory & I/O Tiles
  - No inter-tile cache coherence
- Connected through Network-on-Chip
- Heterogeneous Memory
  - Tile-local Memory
  - Global memory (Off-Tile, via NoC)



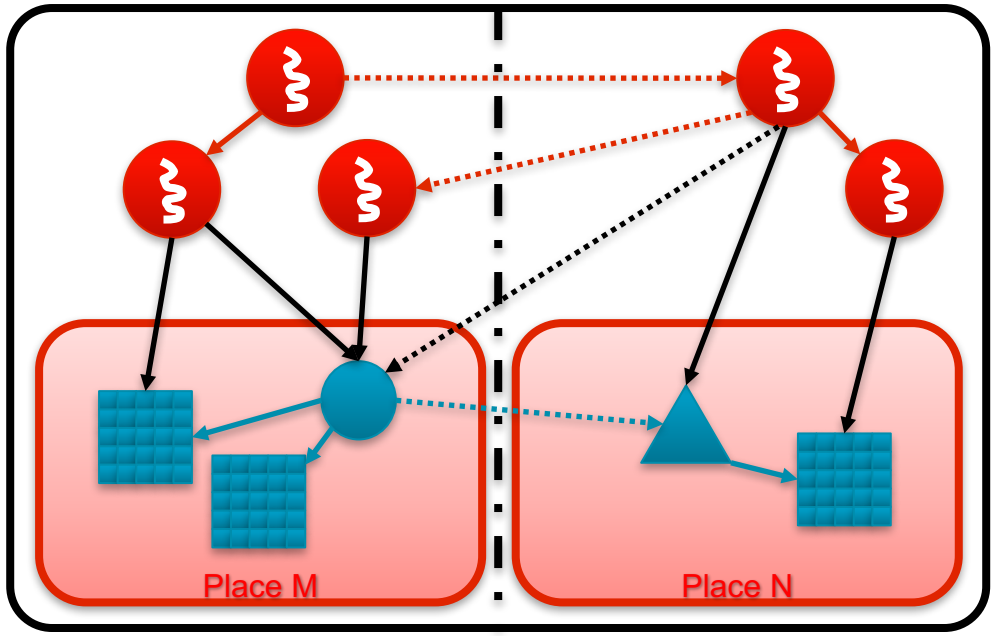
OctoPOS – The Invasive Operating System

- Parallel Operating System tailored for systems with 1000+ cores
- Non-traditional threading scheme: *i*-lets
  - Run-to-Completion semantics with cooperative scheduling
  - Exclusive resource access
  - Binding of *i*-let to execution context only at blocking operations
  - Recycling of execution contexts: Little Overhead for creation, scheduling and dispatch



InvadeX10 – The Invasive Language

- **A**synchronous **P**artitioned **G**lobal **A**ddress **S**pace (APGAS)
  - Activities within an X10 Place may freely access objects allocated by activities spawned in the same Place
  - Global Reference to objects in other places possible
  - Remote objects not accessed directly, instead creation of copies or place-shift
- Natural fit for InvasIC
  - Activities ➤ *i*-lets
  - Places ➤ Tiles
  - Serialization ➤ Direct Cloning
- Invasive Compiler x10i
  - Implements Resource-awareness (invade, infect, retreat)
  - Direct use of OctoPOS APIs
  - Emits Assembly (SPARC, x86)



SWE-X10 – Shallow Water Equations in X10

- Proxy Application for simulation of shallow water waves
- Compute propagation of tsunamis given initial displacement
- Simulate inundation of coastal areas

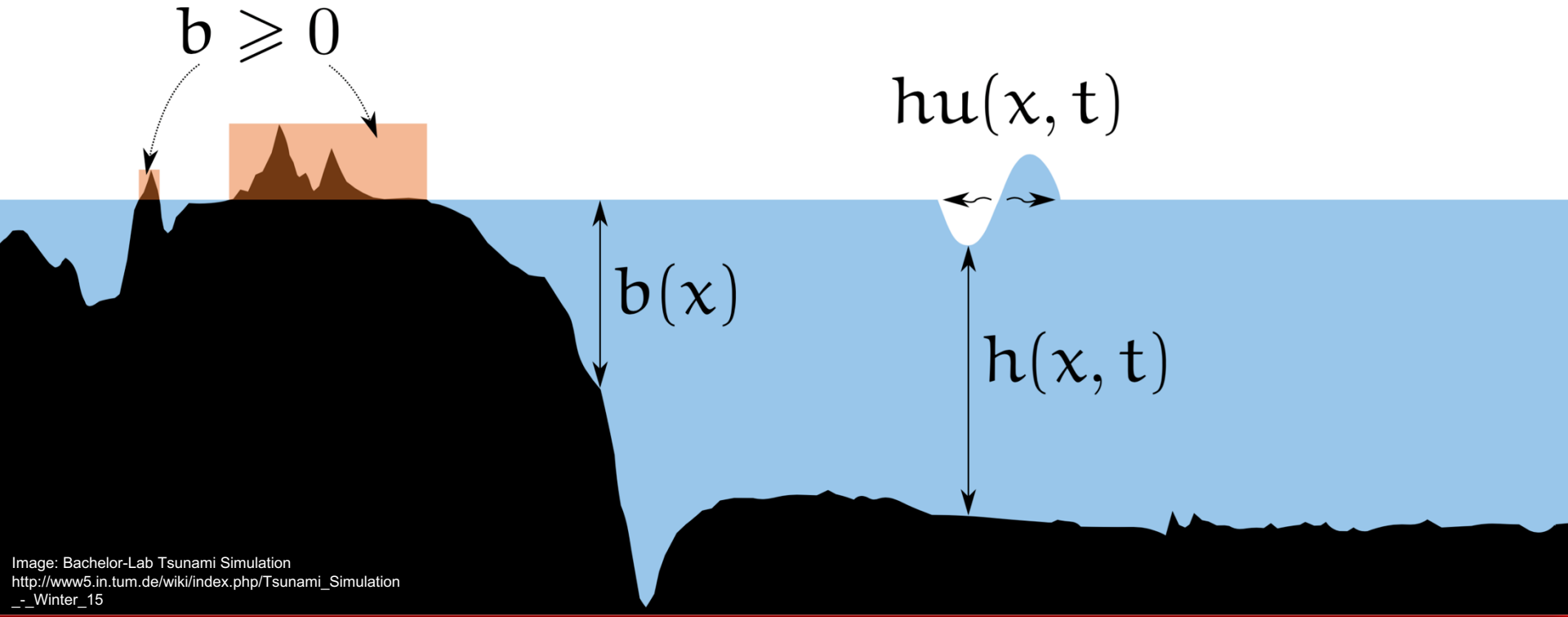


Image: Bachelor-Lab Tsunami Simulation
http://www5.in.tum.de/wiki/index.php/Tsunami_Simulation_-_Winter_15

SWE-X10 - Discretization

- Finite volume scheme on a Cartesian grid with piecewise constant unknown quantities and Eulerian time step

$$Q_{i,j}^{n+1} = Q_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-\frac{1}{2},j} + \mathcal{A}^{-}\Delta Q_{i+\frac{1}{2},j}^{n}\right) - \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-\frac{1}{2}} + \mathcal{B}^{-}\Delta Q_{i,j+\frac{1}{2}}^{n}\right)$$

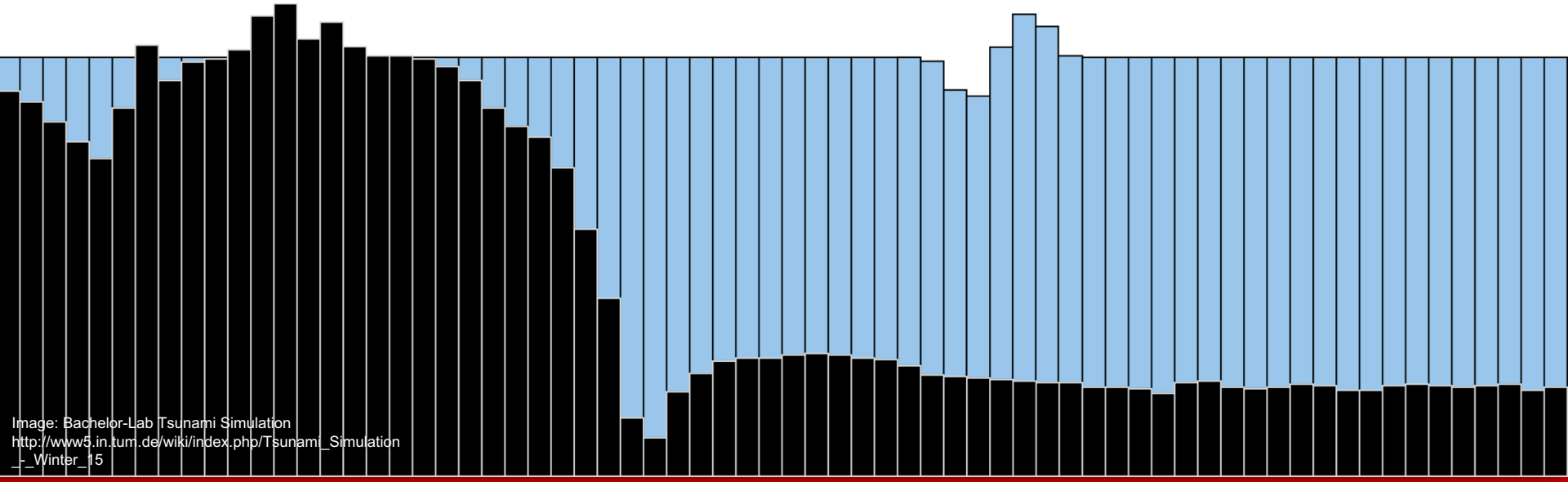
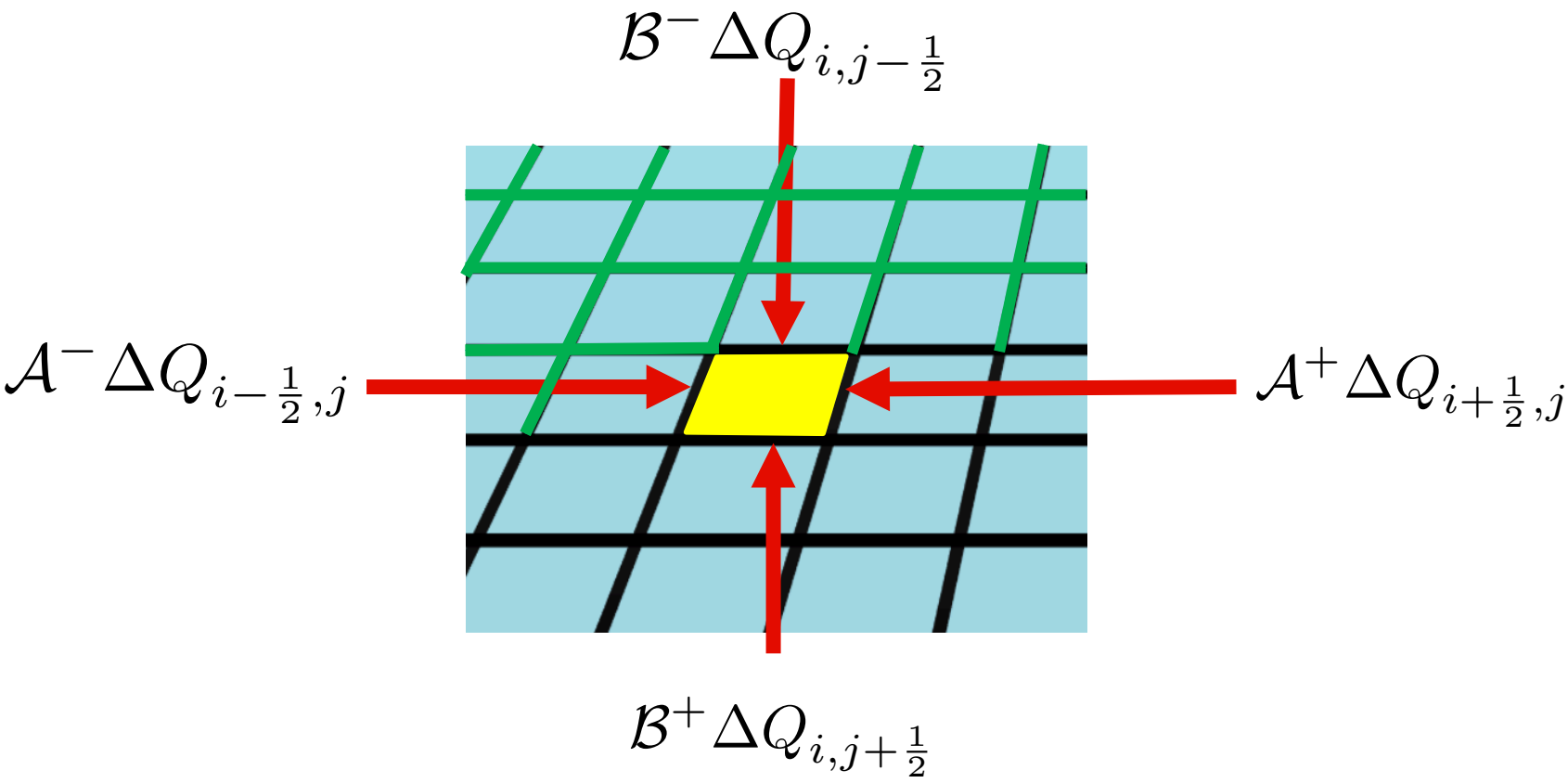


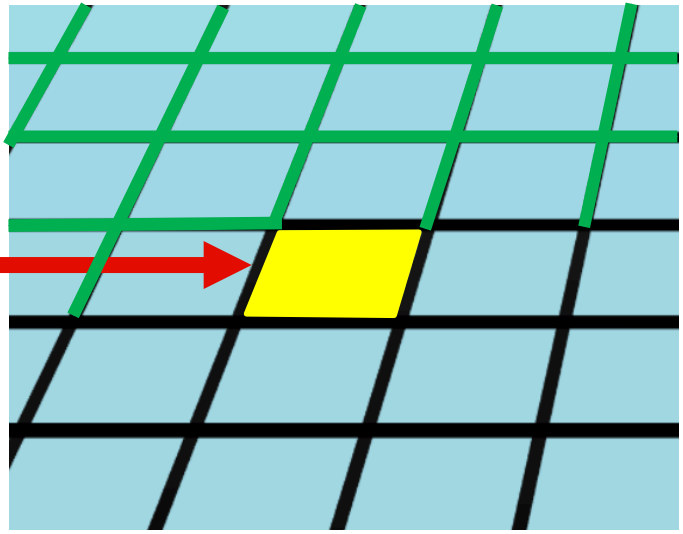
Image: Bachelor-Lab Tsunami Simulation
http://www5.in.tum.de/wiki/index.php/Tsunami_Simulation_-_Winter_15

SWE-X10 - Discretization

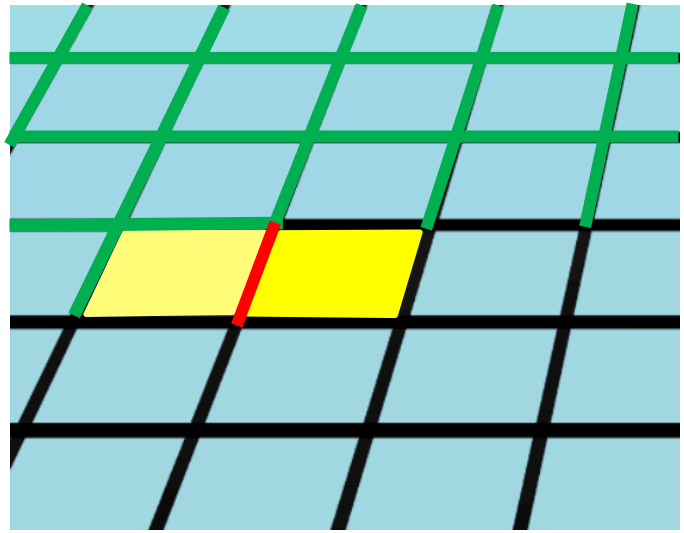


SWE-X10 - Discretization

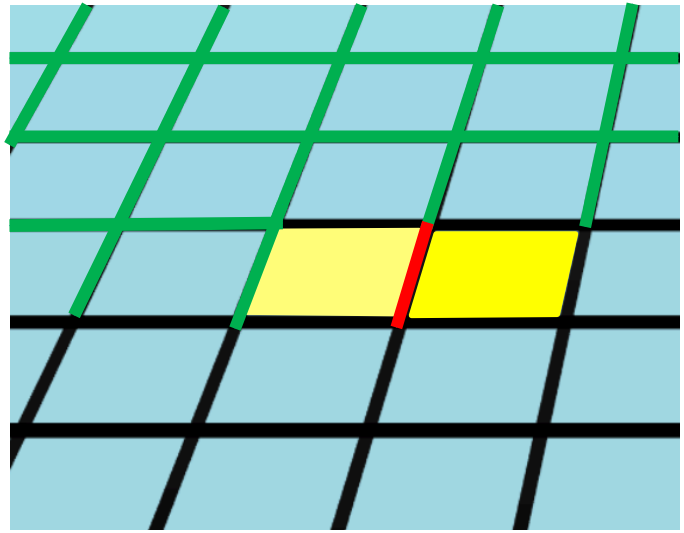
$$\mathcal{A}^- \Delta Q_{i-\frac{1}{2},j}$$



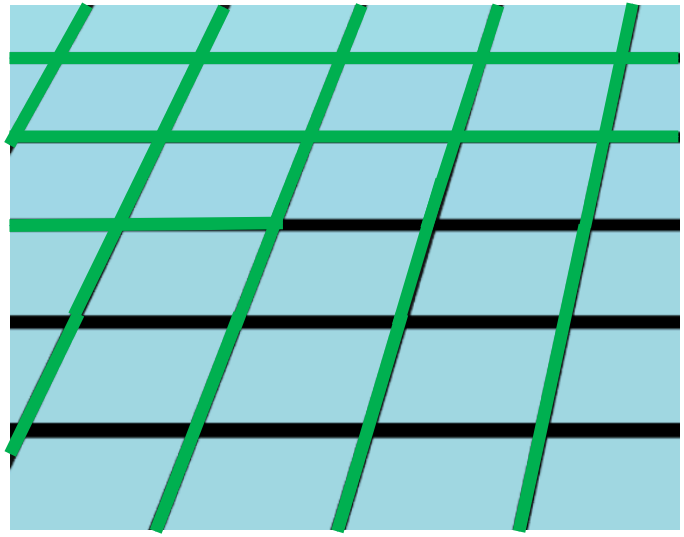
SWE-X10 - Discretization



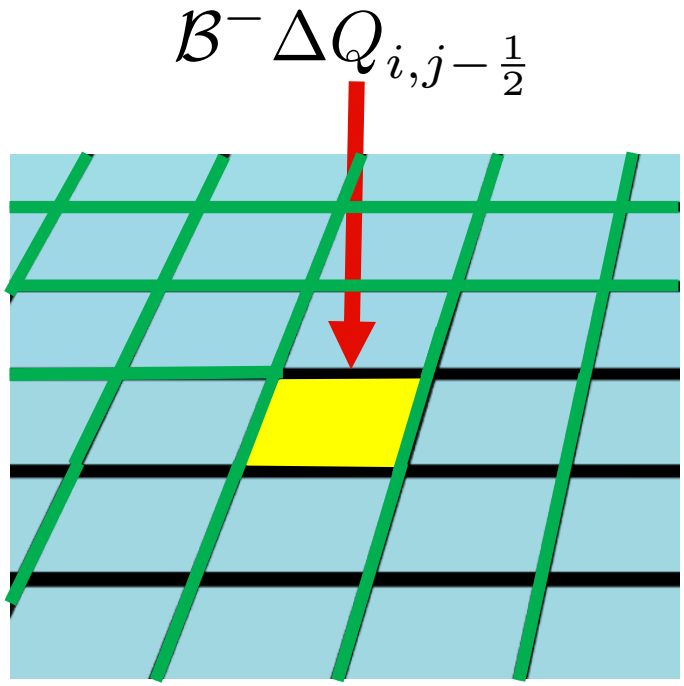
SWE-X10 - Discretization



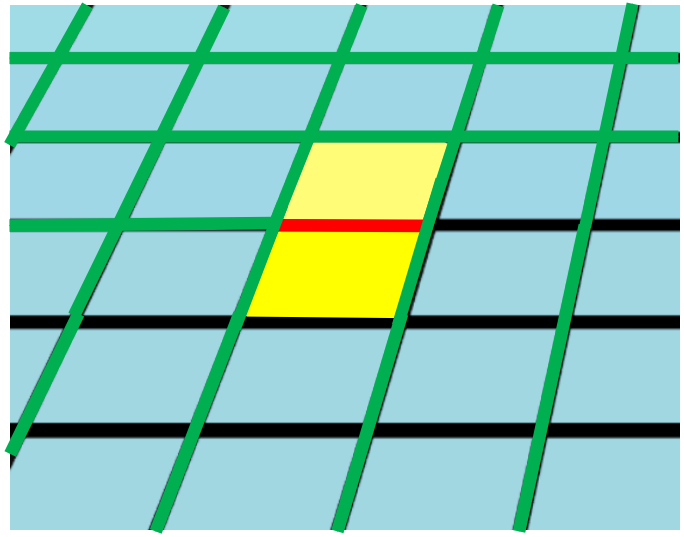
SWE-X10 - Discretization



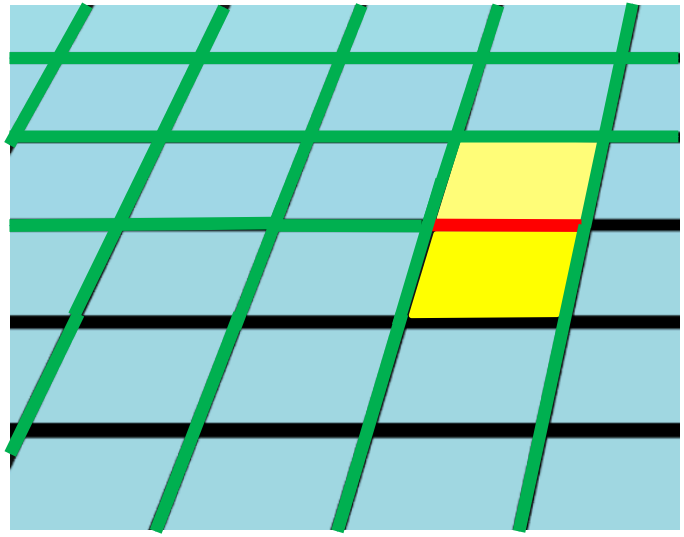
SWE-X10 - Discretization



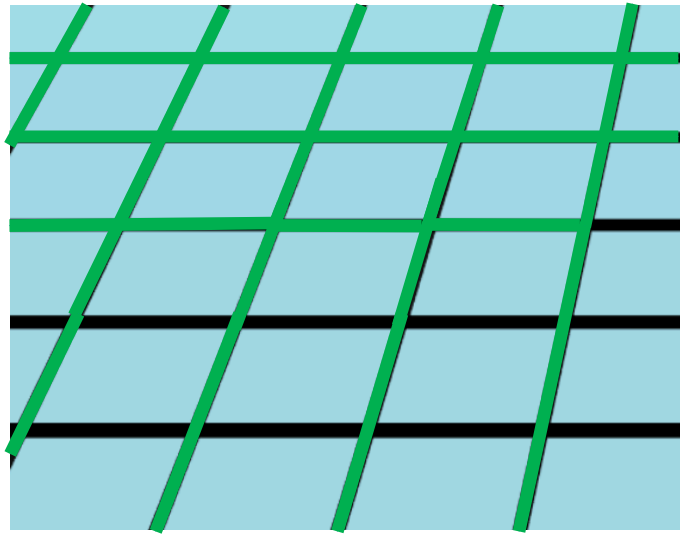
SWE-X10 - Discretization



SWE-X10 - Discretization

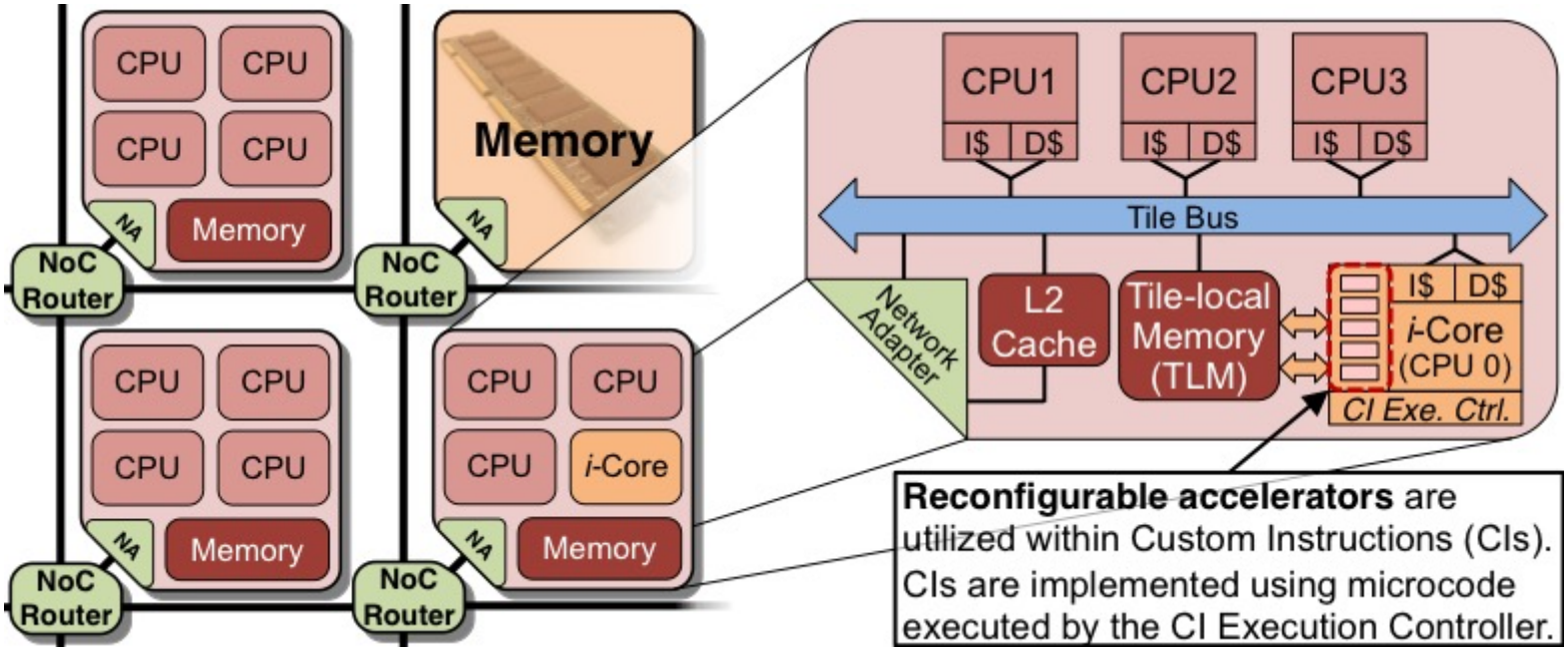


SWE-X10 - Discretization



i-Core

- Combination of “normal” CPU core and application-specific accelerators (through FPGA fabric)
  - Realized through Custom Instructions (CI)
  - May be loaded at run time by application



Acceleration of SWE-X10 using *i*-Core

- Custom Instruction for computation of approximate solutions of Riemann Problems (f-Wave solver)
- Pipelined Accelerators (for operations used in solver):
  - FP_MAC (3-5cy),
  - FP_DIV (6 cy),
  - FP_SQRT (5 cy),
  - FP_UTIL (3 cy)
- Performs all 54 floating point operations as single CI
  - Data-flow graph with 97 nodes/operations
  - 5 accelerators used: 2x FP_MAC, 1x FP_DIV, 1x FP_SQRT and 1x FP_UTIL
- Configuration at application startup

Adaptions in SWE-X10

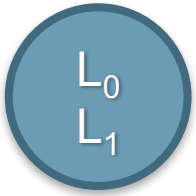
Previous

Current

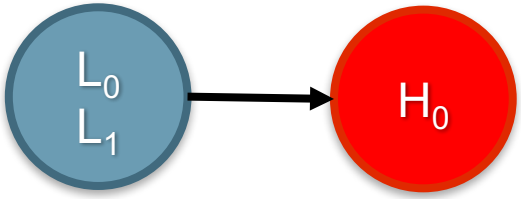
Next

Tile-Local Memory

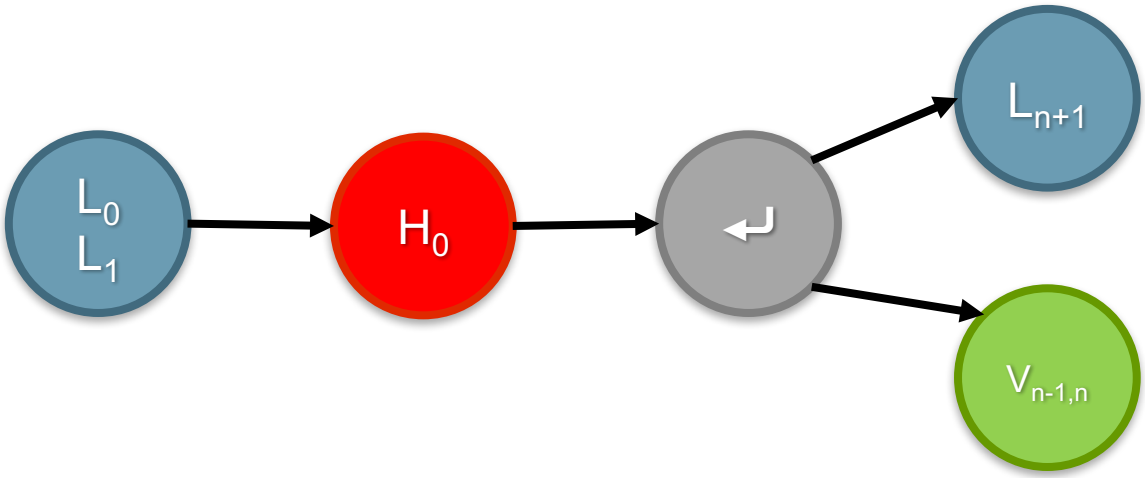
Adaptions in SWE-X10

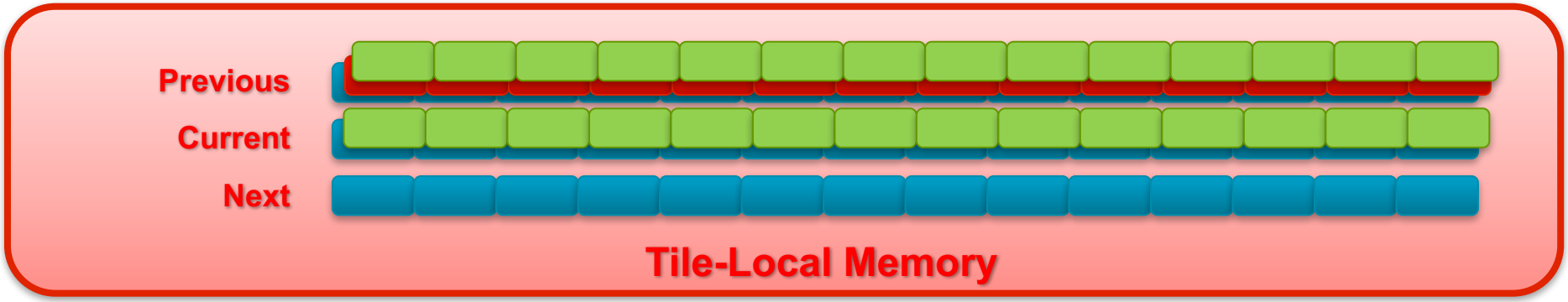


Adaptions in SWE-X10

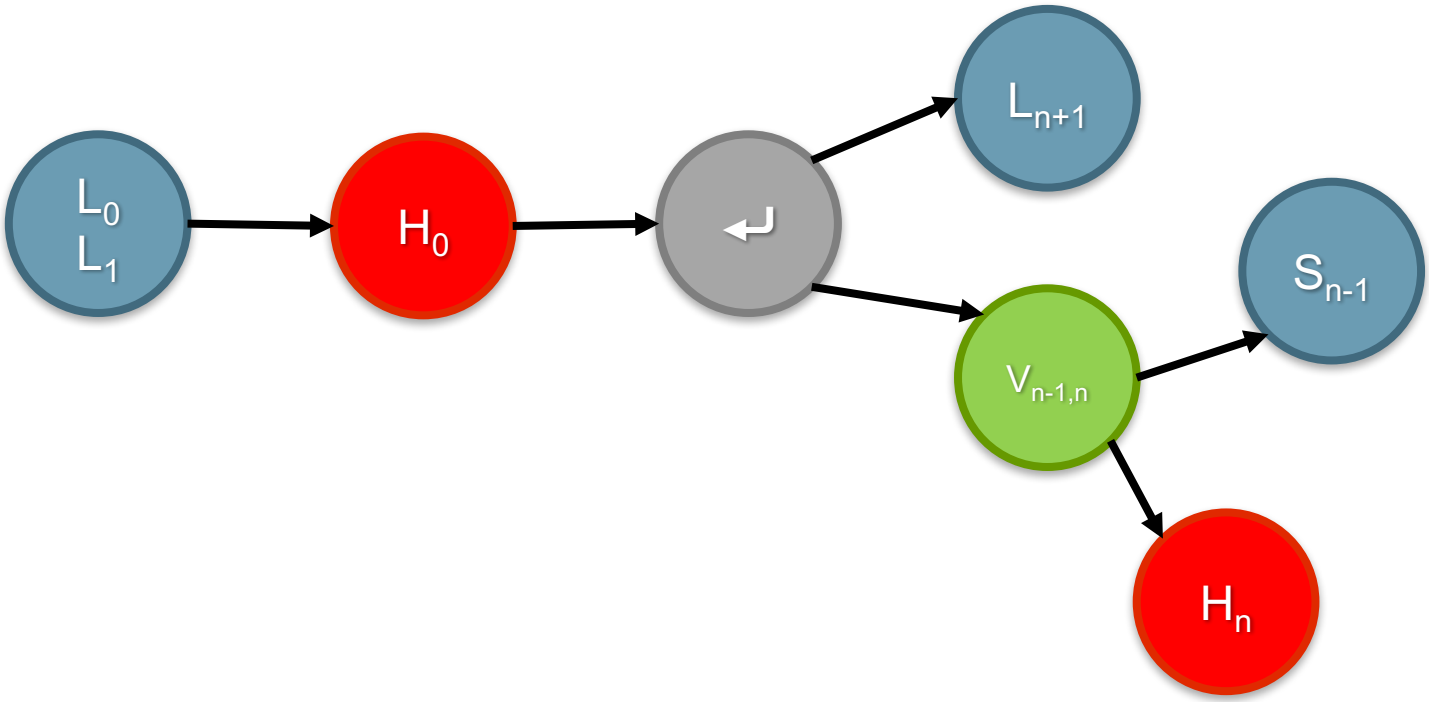


Adaptions in SWE-X10

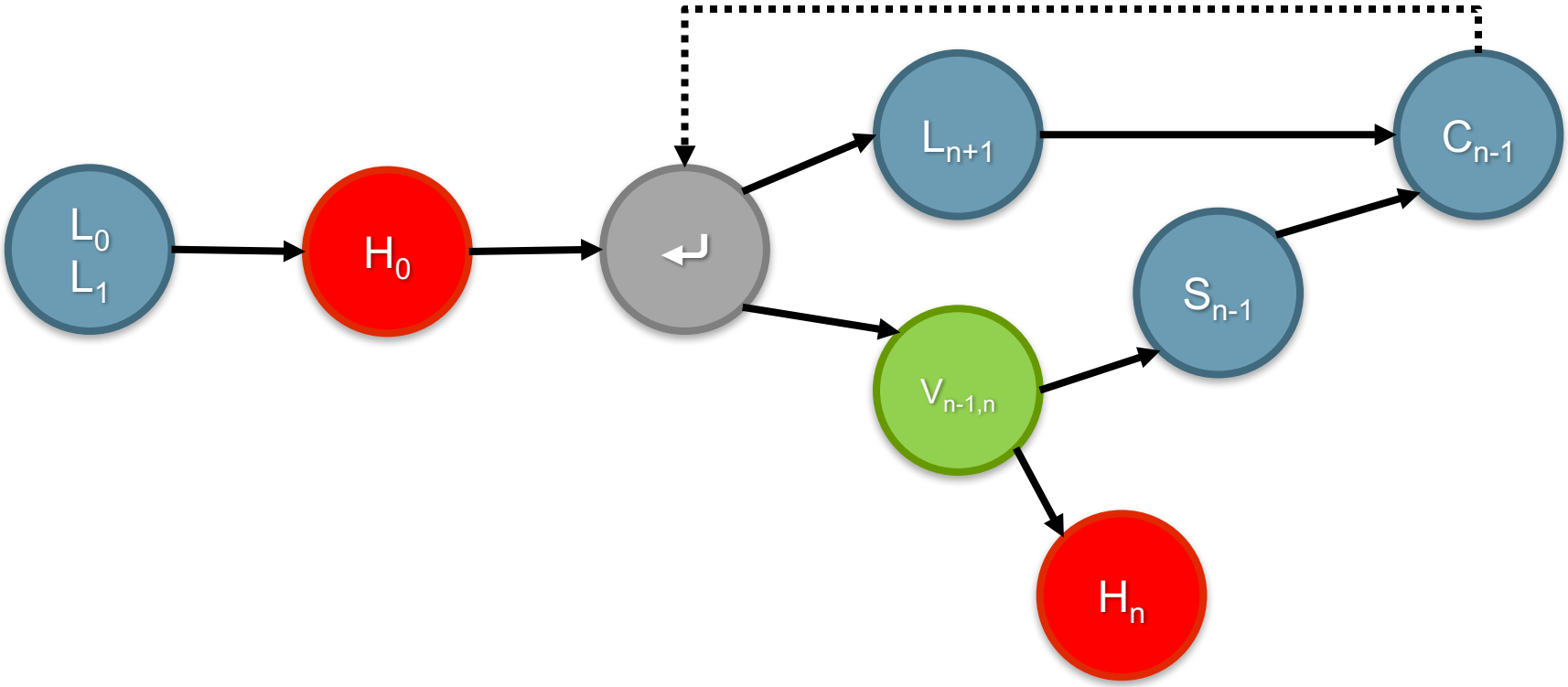




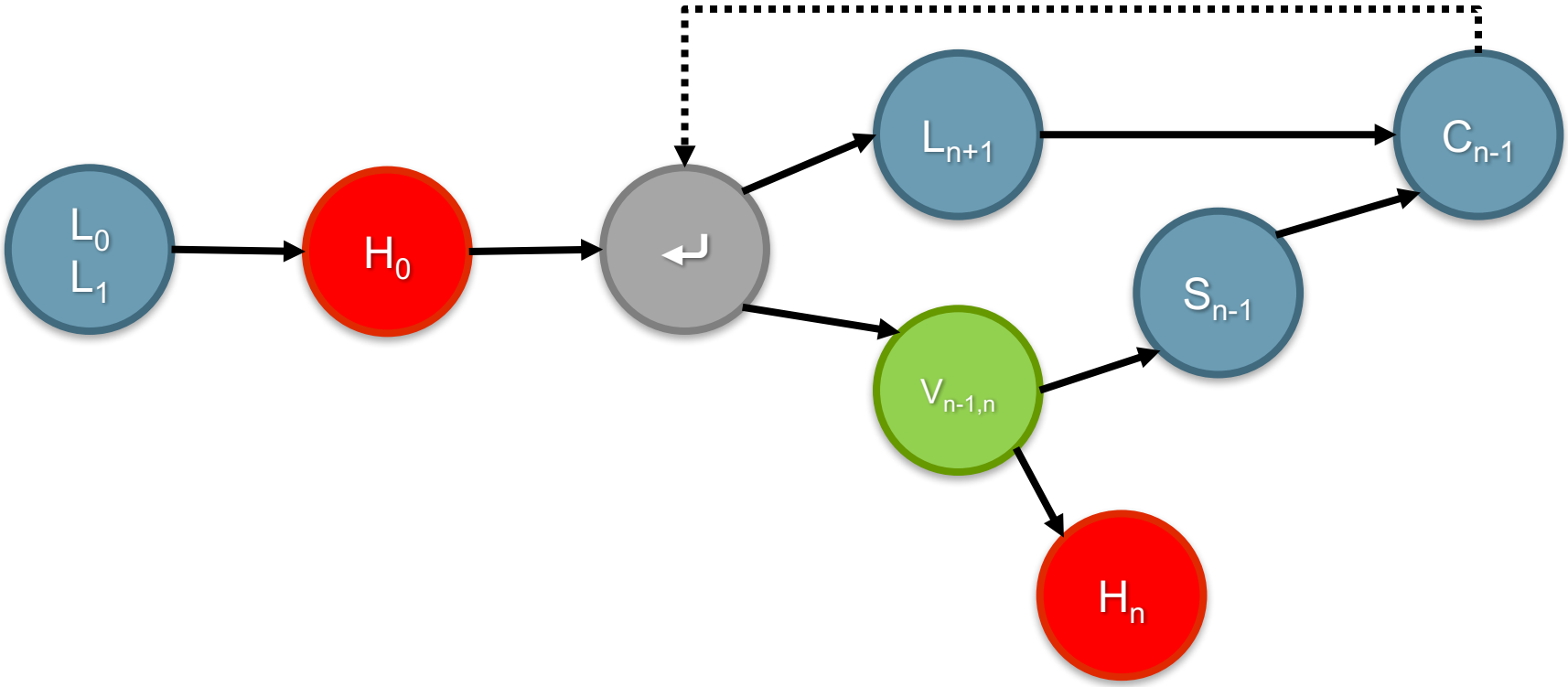
Adaptions in SWE-X10

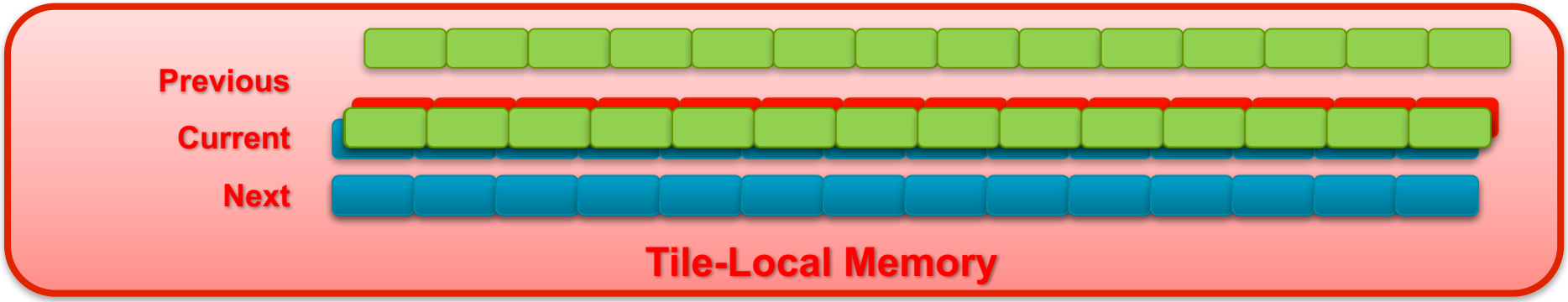


Adaptions in SWE-X10

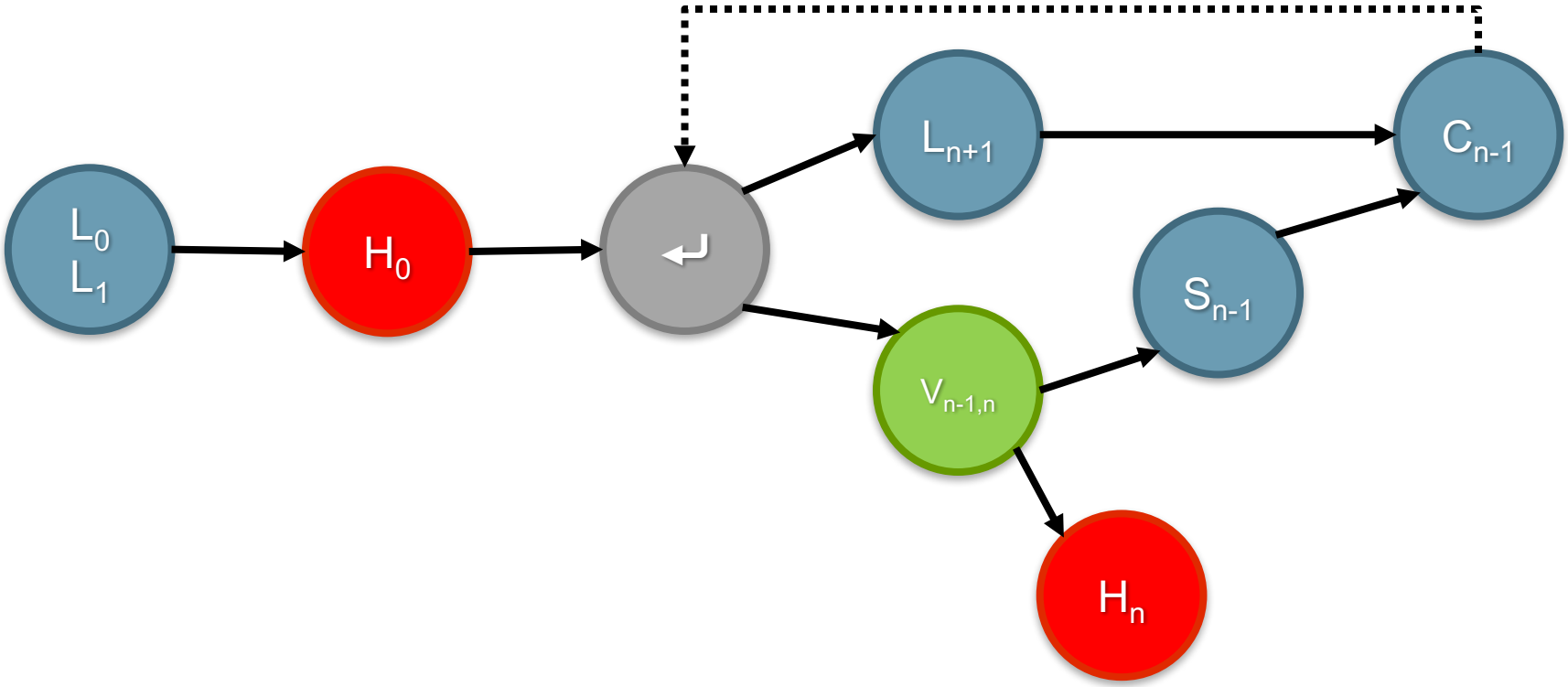


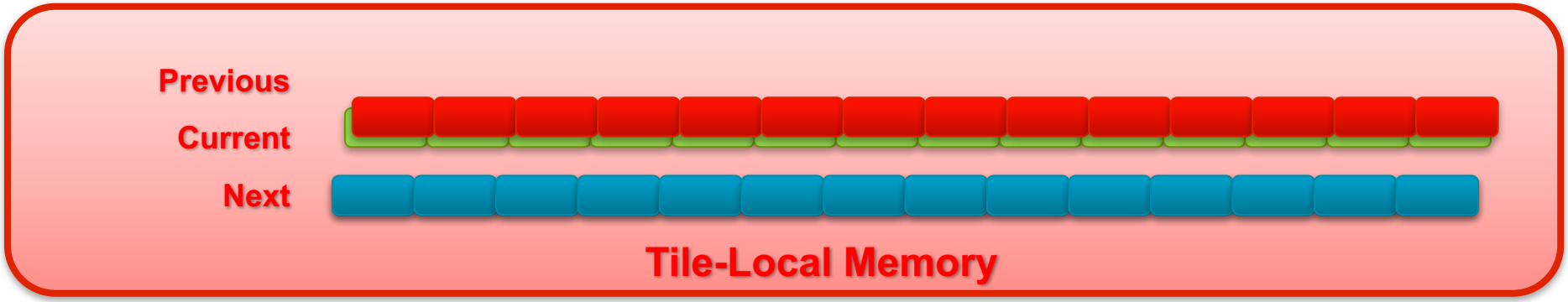
Adaptions in SWE-X10



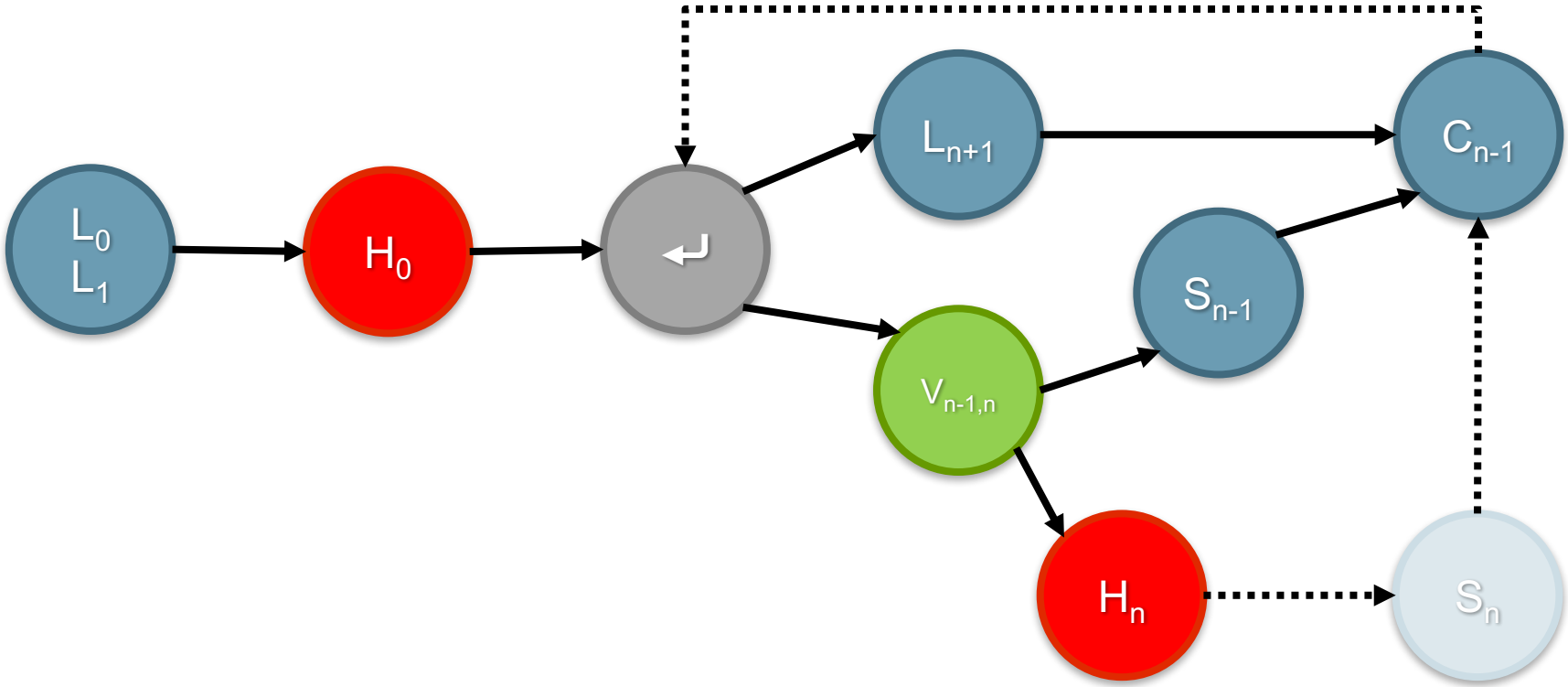


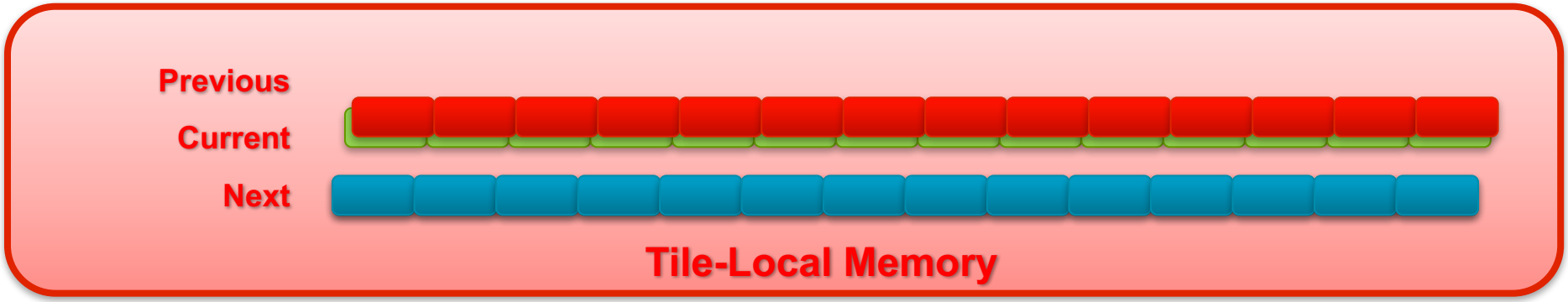
Adaptions in SWE-X10





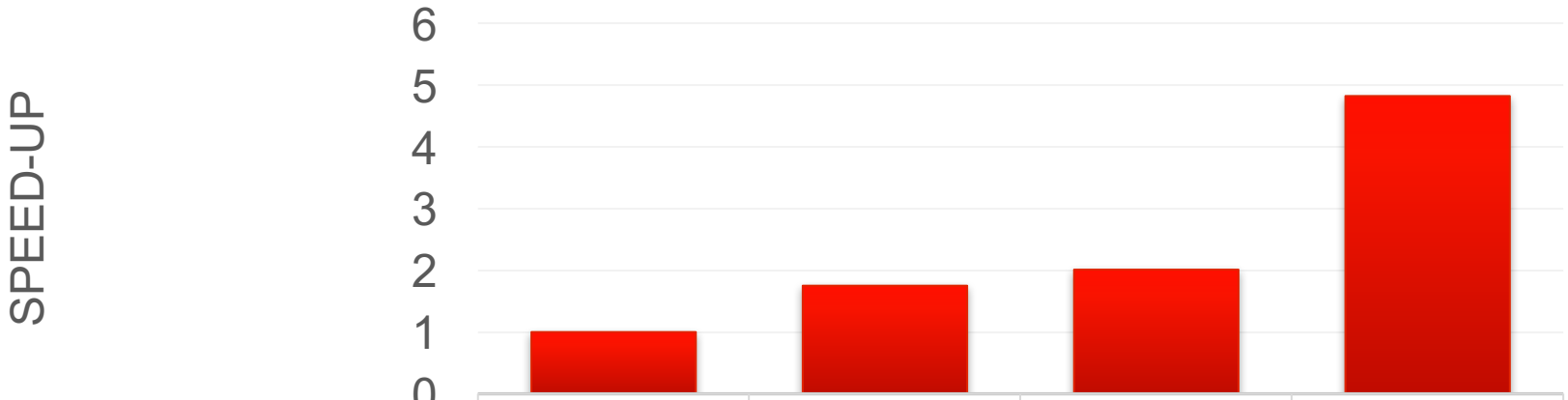
Adaptions in SWE-X10





Evaluation

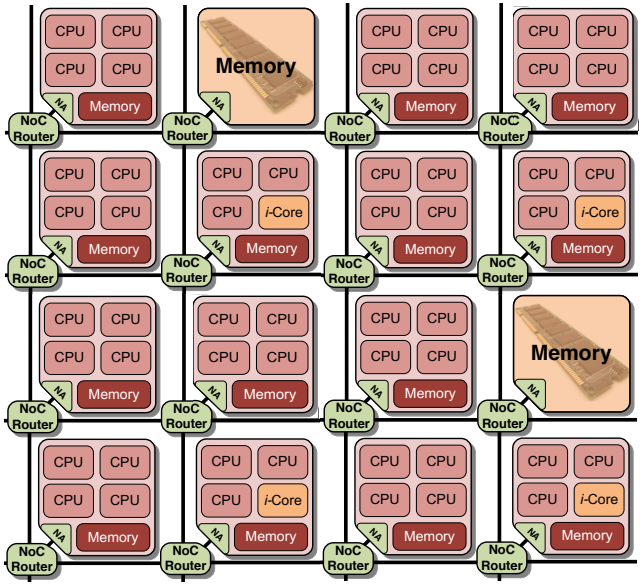
- Two Potential sources of performance gain:
  - Tile-local memory
  - *i*-Core
- Single iteration on one patch with 60x60 grid cells

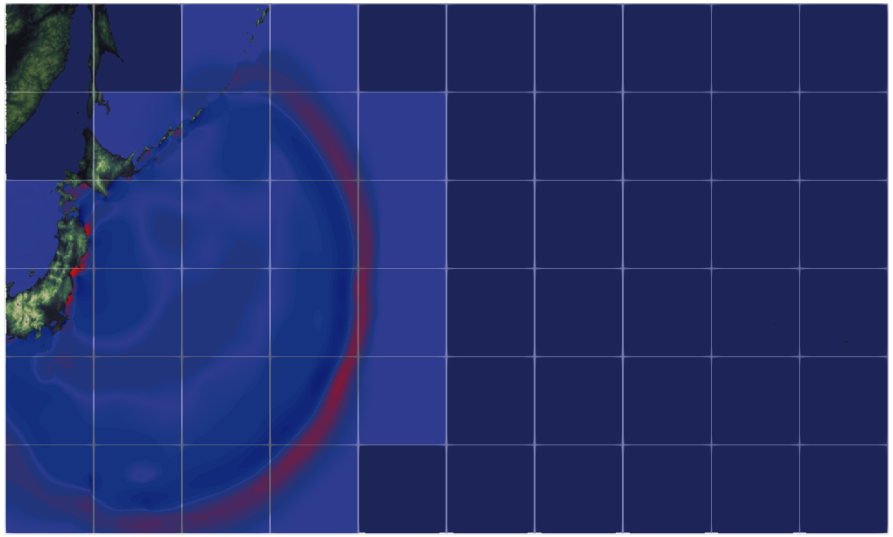


	LEON GM	LEON TLM	i-Core GM	i-Core TLM
Speedup	1	1,75	2,01	4,82
Execution time [ms]	2049	1169	1017	425

Outlook

- Model HLLE Riemann solver – enable coastal flooding
- Evaluate whole-system performance benefits
- Scale to and evaluate with larger hardware configuration (e.g. 4x4 tiles with ~64 cores multiple *i*-Cores)





Questions?

Acknowledgements

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).