Scalable Multi-FPGA Design of a Discontinuous Galerkin Shallow-Water Model on Unstructured Meshes

Jennifer Faj[1], Tobias Kenter[2], Sara Faghih-Naini[3],
Christian Plessl[2], Vadym Aizinger[3]

Paderborn University, Germany
Paderborn Center for Parallel Computing

[1]Paderborn University, now KTH Stockholm
[2]Paderborn University, [3]University of Bayreuth

Paderborn
Center for
Parallel
Computing

Overview

- PASC 2021: first FPGA implementation of SWS DG code [1]
  - single FPGA (Stratix 10)
  - fast: up to 717 GFLOPS
  - unstructured mesh in on-chip memory -> limited problem sizes

- PASC 2023: partitioned multi-FPGA design
  - up to 10 directly communicating FPGAs (Stratix 10)
  - faster: up to 6.5 TFLOPS
  - scalable problem sizes (weak + strong scaling)

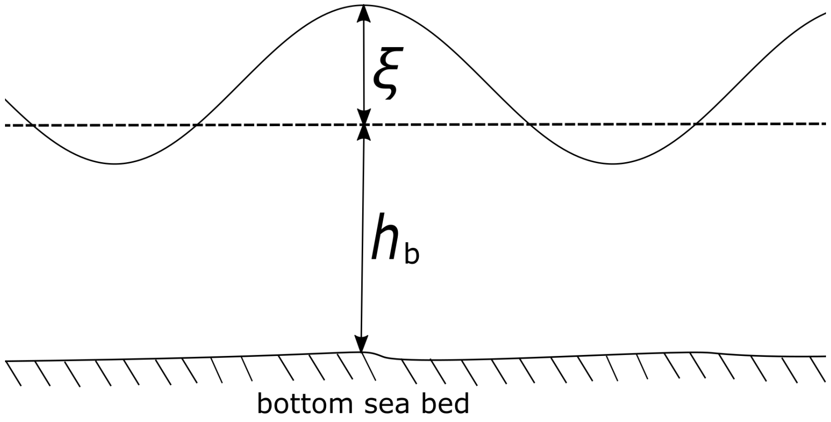
[1] **Algorithm-Hardware Co-design of a Discontinuous Galerkin Shallow-Water Model for a Dataflow Architecture on FPGA.** Kenter, Shambhu, Faghih-Naini, Aizinger, PASC'21

Shallow Water DG Code

Shallow Water Equations

- 2D shallow water equations (SWE) (derived from the Navier-Stokes equations)

  - $\partial_t \xi + \nabla \cdot \mathbf{u} = 0$
  - $\partial_t \mathbf{u} + \nabla \cdot \left(\frac{\mathbf{u} \otimes \mathbf{u}}{\mathrm{H}}\right) + \tau_{bf}\mathbf{u} + f_{\mathrm{c}}\mathbf{k} \times \mathbf{u} + \mathrm{gH}\nabla\xi = \mathbf{F}$



with unknowns

ξ : elevation of free water surface, $\mathbf{u} = (U, V)^T$: depth integrated horizontal velocity field

and parameters

h_b: bathymetric depth, $\mathrm{H} = h_b + \xi$: total fluid depth, τ_{bf} : bottom friction coefficient

f_c : Coriolis coefficient, $\mathbf{k}$: unit vertical vector, g : gravitational acceleration

F : forcing term from wind and atmospheric pressure gradient

DG Formulation

- **Uses Discontinuous Galerkin method on unstructured triangular meshes**

$$\int_{\Omega_i} \partial_t \mathbf{c}_\Delta \boldsymbol{\varphi}\, dx + \underset{\text{Edge kernel}}{\boxed{\int_{\partial\Omega_i} \widehat{\mathrm{A}}(\boldsymbol{c}_\Delta, {\boldsymbol{c}_\Delta}^{+}, \boldsymbol{n})\, \boldsymbol{\varphi}\, ds}} - \underset{\text{Element kernel}}{\boxed{\int_{\Omega_i} \mathrm{A}(\boldsymbol{c}_\Delta)\ \cdot\ \nabla \varphi\, dx = \int_{\Omega_i} \boldsymbol{r}(\boldsymbol{c}_\Delta)\, \boldsymbol{\varphi}\, dx}}$$

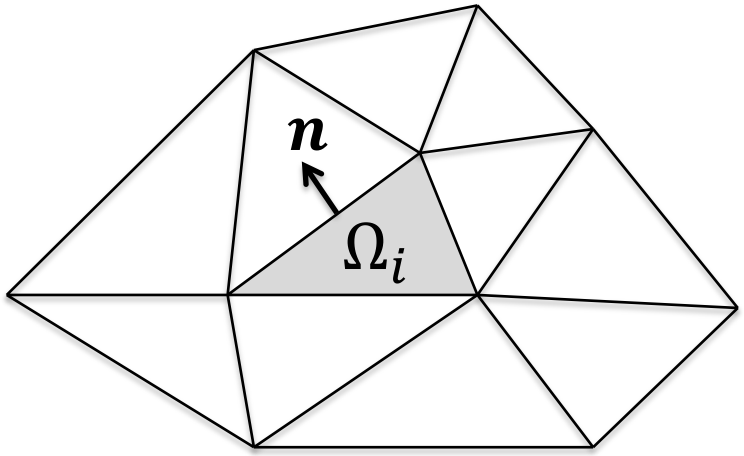
where

$\mathbf{c}_\Delta = (\xi_\Delta, \mathrm{U}_\Delta, \mathrm{V}_\Delta)^T$: the discrete vector of unknowns restricted to Ω_i,

${\mathbf{c}_\Delta}^{+}$: the discrete vector of unknowns restricted to the edge-neighbour of Ω_i,

$\boldsymbol{n}$: the exterior unit normal to $\partial\Omega_i$, $\boldsymbol{\varphi}$: test function

$\widehat{\mathrm{A}}$: numerical flux from Riemann solver (Lax-Friedrichs)



UTBEST Overview

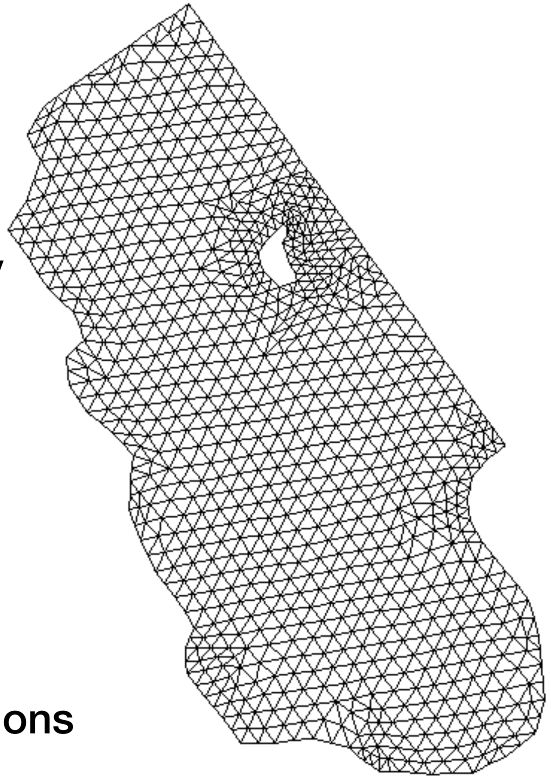
- I/O and grid management: FORTRAN
- DG scheme + computationally intensive parts: C
  - works in single precision
- 3 polynomial DG discretizations
  - piecewise constant (PC) (= cell-centered finite volumes)
  - piecewise linear (PL)
  - piecewise quadratic (PQ)
- Integration kernels
  - elements: 1, 4, 9 quadrature points
  - edges: 1, 2, 3 quadrature points
    - Lax-Friedrichs Riemann solver
- Corresponding time discretization
  - Runge-Kutta orders 1, 2, 3

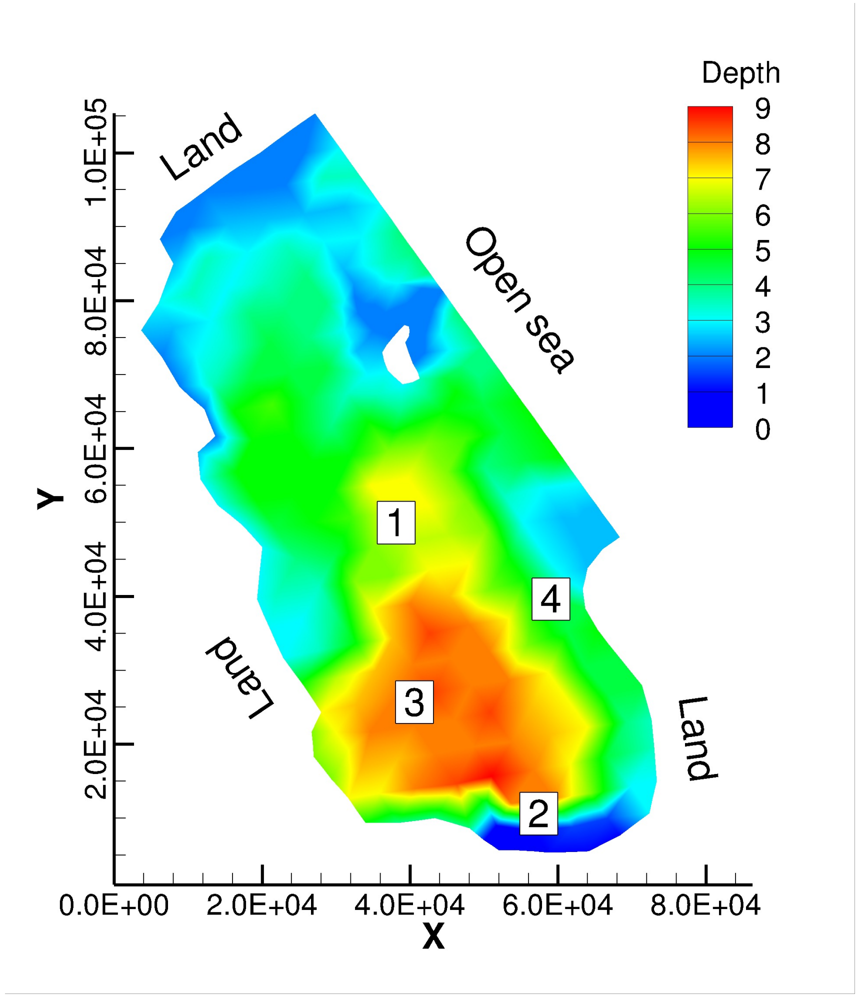
$$\int_{\Omega_i} \partial_t \mathbf{c}_\Delta \boldsymbol{\varphi}\, dx + \boxed{\int_{\partial\Omega_i} \widehat{\mathrm{A}}(\boldsymbol{c}_\Delta, \boldsymbol{c}_\Delta{}^{+}, \boldsymbol{n})\, \boldsymbol{\varphi}\, ds} - \boxed{\int_{\Omega_i} \mathrm{A}(\boldsymbol{c}_\Delta) \cdot \nabla\varphi\, dx = \int_{\Omega_i} \boldsymbol{r}(\boldsymbol{c}_\Delta)\, \boldsymbol{\varphi}\, dx}$$

[V. Aizinger and C. Dawson. 2002. Adv. in Water Resources 25, 1]

Benchmark Scenario

- Bahamas (Bight of Abaco)
  - unstructured mesh
    - 1696 elements
    - h-refinement with factors 2^n
  - tidal forcing at open sea boundary
  - benchmark runs
    - simulated 1 day
    - time step 5s
    - 17280 steps
  - outputs
    - elevation snapshots
    - full time series at observation stations

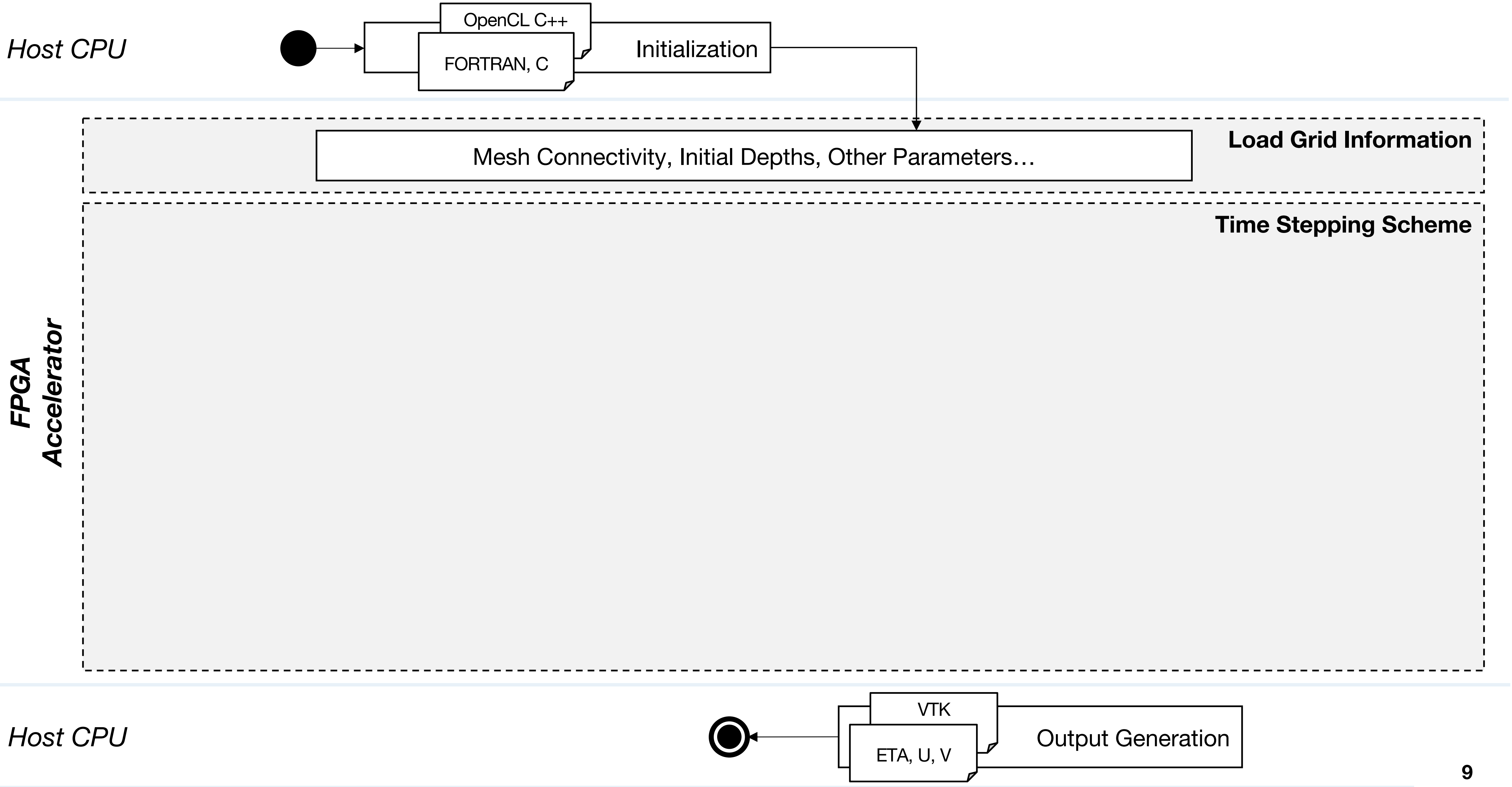




bathymetry + observation stations

Single FPGA Design

UTBEST2D Single FPGA Design [1]



Host CPU

OpenCL C++

FORTRAN, C

Initialization

FPGA Accelerator

Load Grid Information

Mesh Connectivity, Initial Depths, Other Parameters…

Time Stepping Scheme

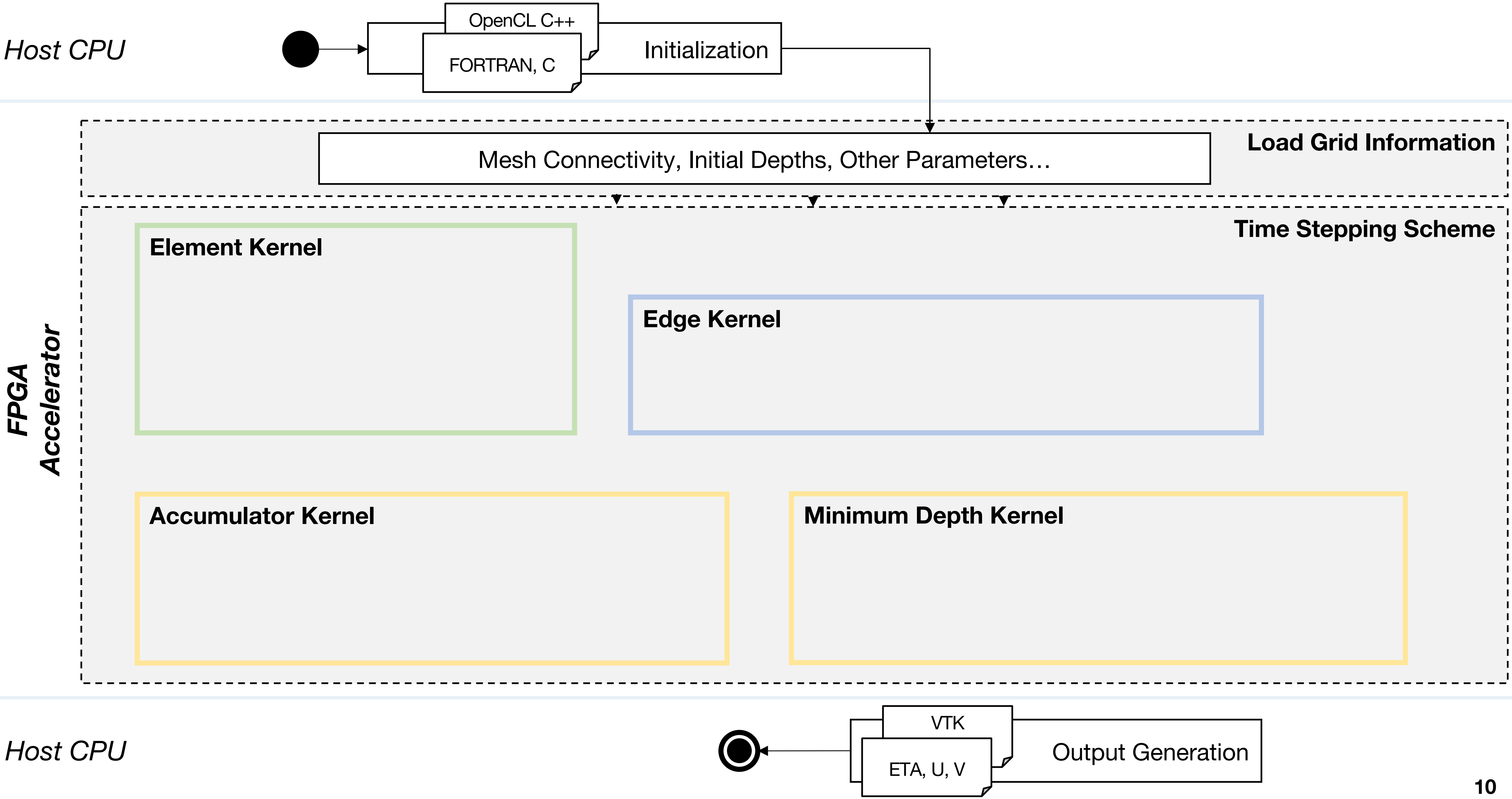
Host CPU

VTK

ETA, U, V

Output Generation

UTBEST2D Single FPGA Design [1]



Host CPU

OpenCL C++

FORTRAN, C

Initialization

Load Grid Information

Mesh Connectivity, Initial Depths, Other Parameters…

FPGA Accelerator

Time Stepping Scheme

Element Kernel

Edge Kernel

Accumulator Kernel

Minimum Depth Kernel

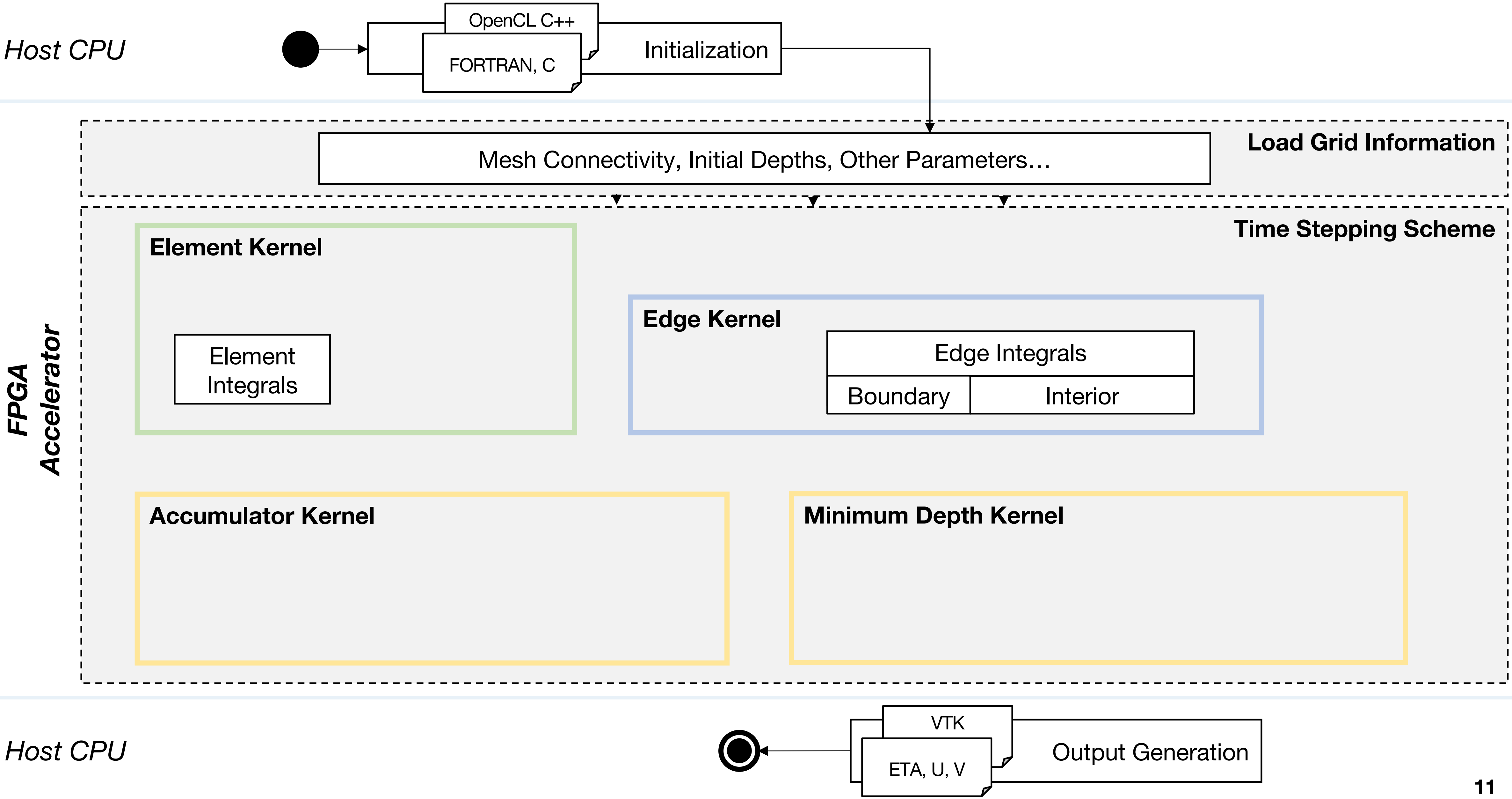
Host CPU

VTK

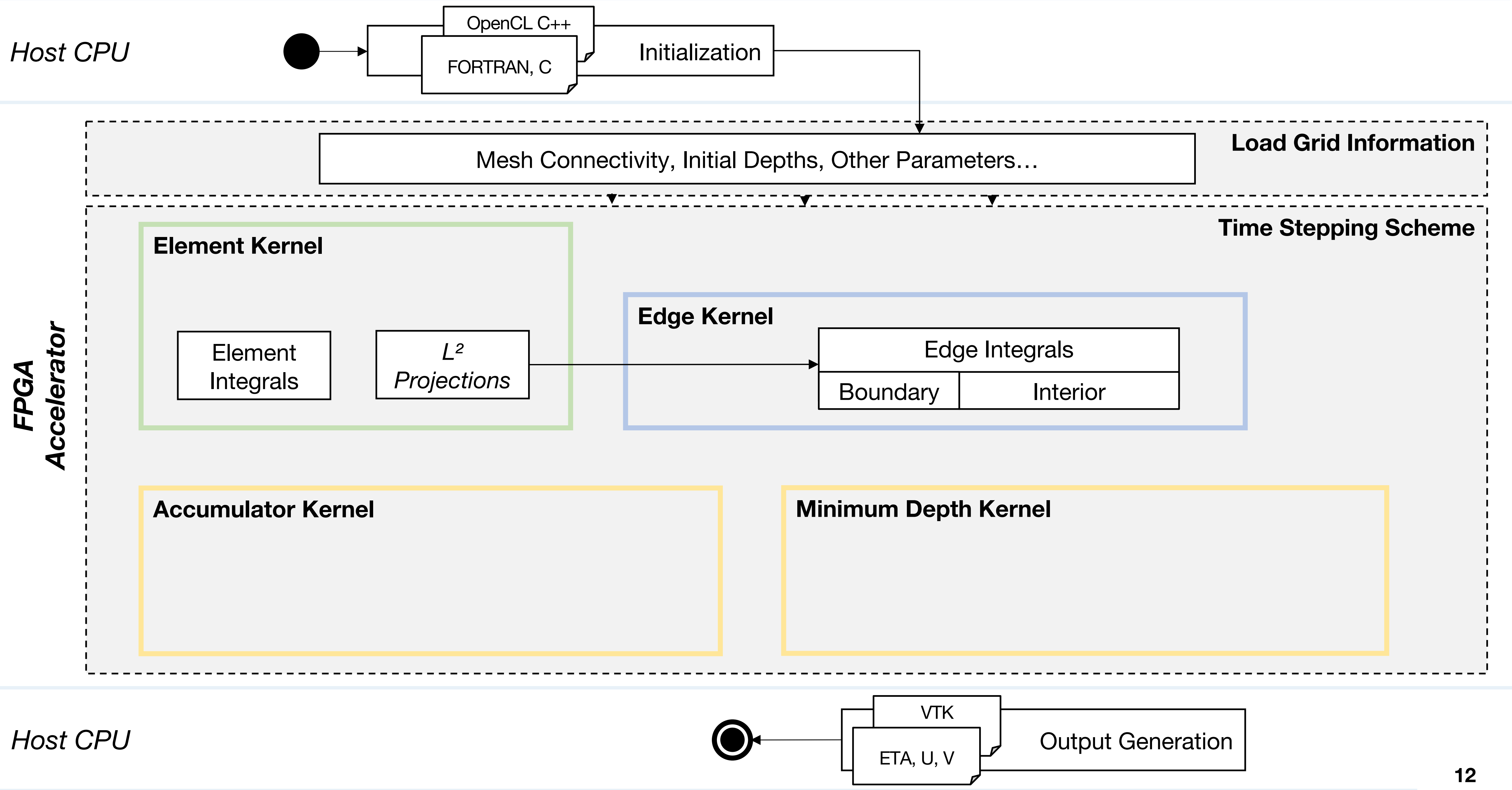
ETA, U, V

Output Generation

UTBEST2D Single FPGA Design [1]



UTBEST2D Single FPGA Design [1]



Host CPU

OpenCL C++

FORTRAN, C

Initialization

Load Grid Information

Mesh Connectivity, Initial Depths, Other Parameters…

FPGA Accelerator

Time Stepping Scheme

Element Kernel

Element Integrals

L^2 Projections

Edge Kernel

Edge Integrals

Boundary

Interior

Accumulator Kernel

Minimum Depth Kernel

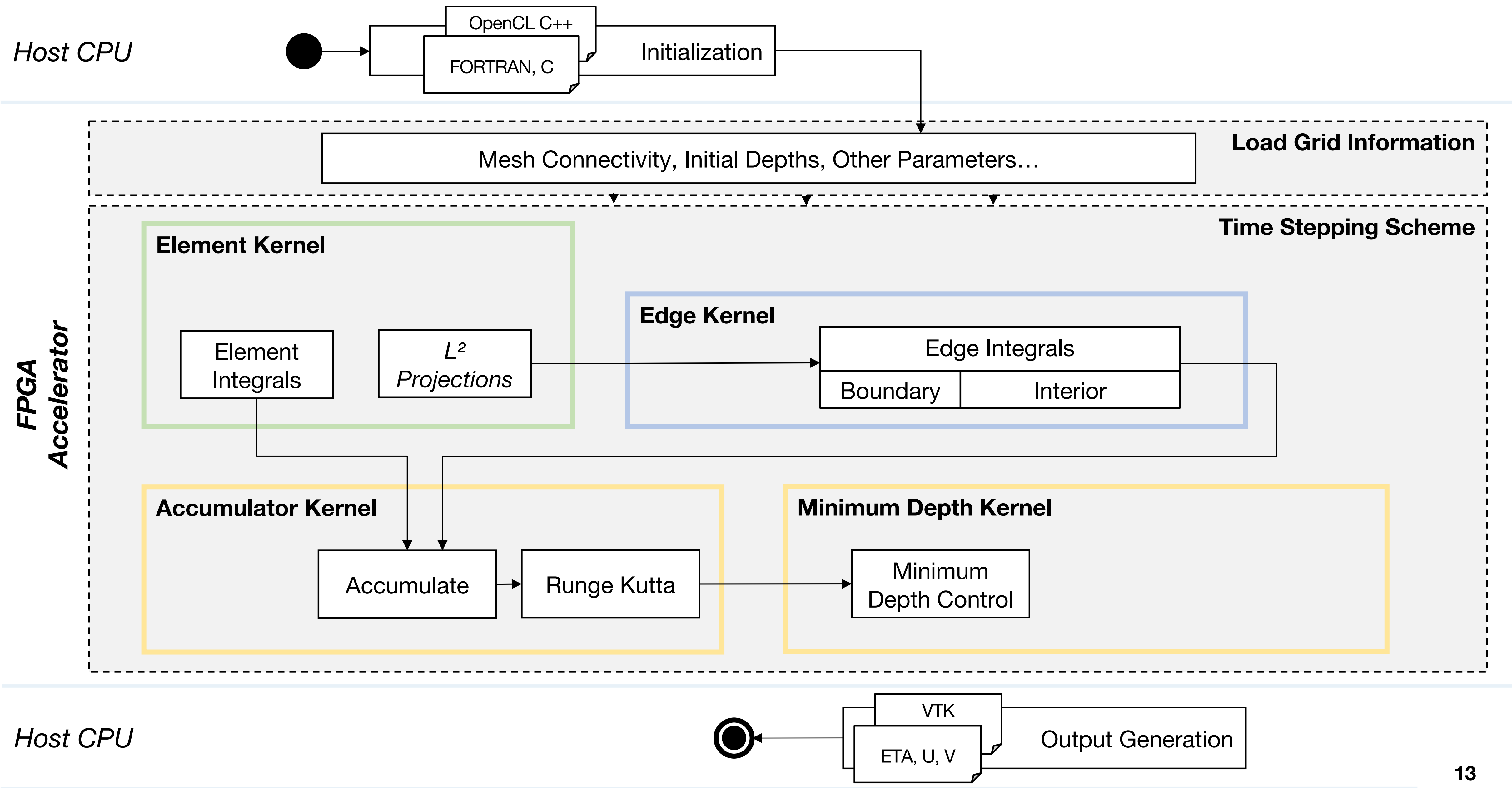
Host CPU

VTK

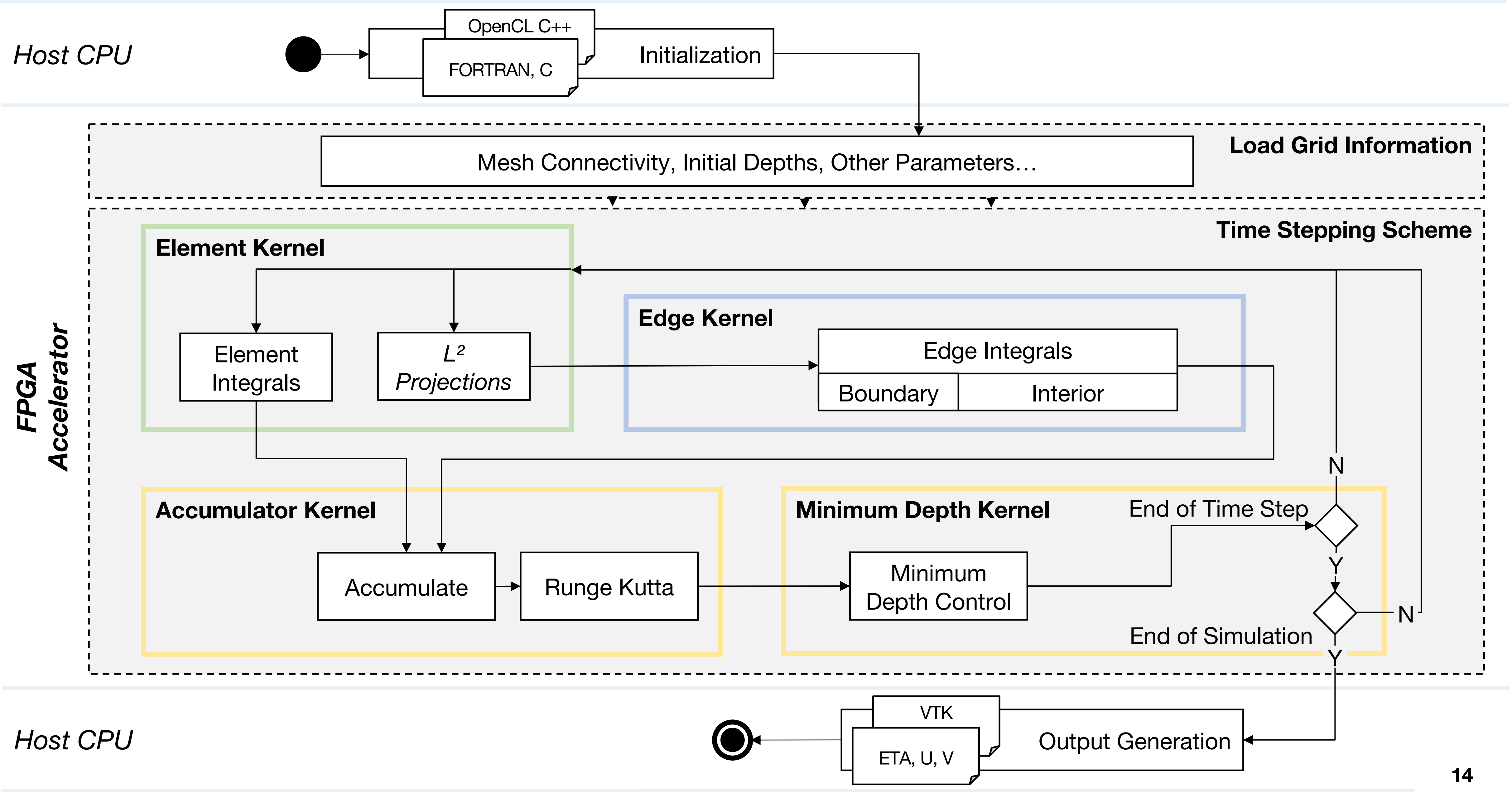
ETA, U, V

Output Generation

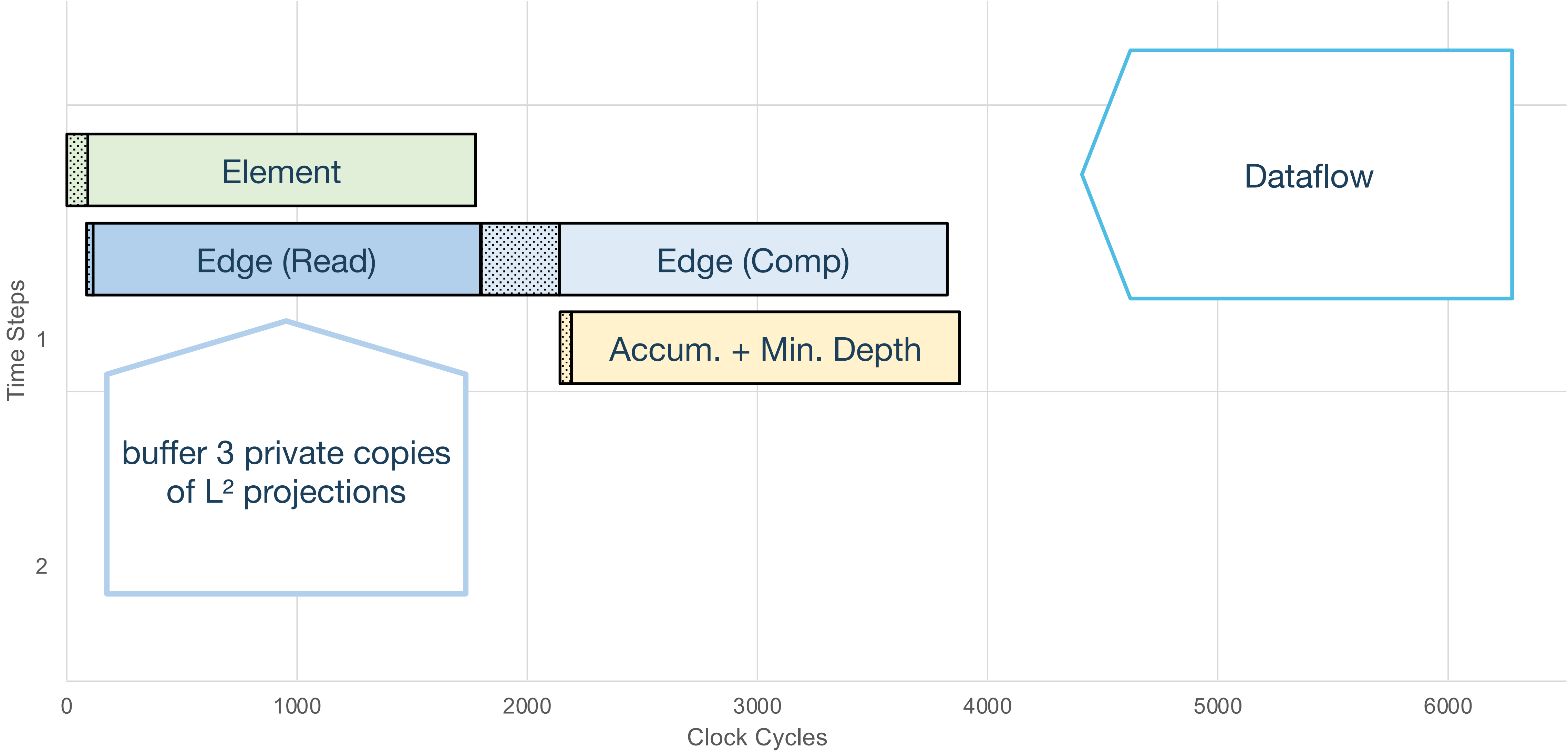
UTBEST2D Single FPGA Design [1]



UTBEST2D Single FPGA Design [1]

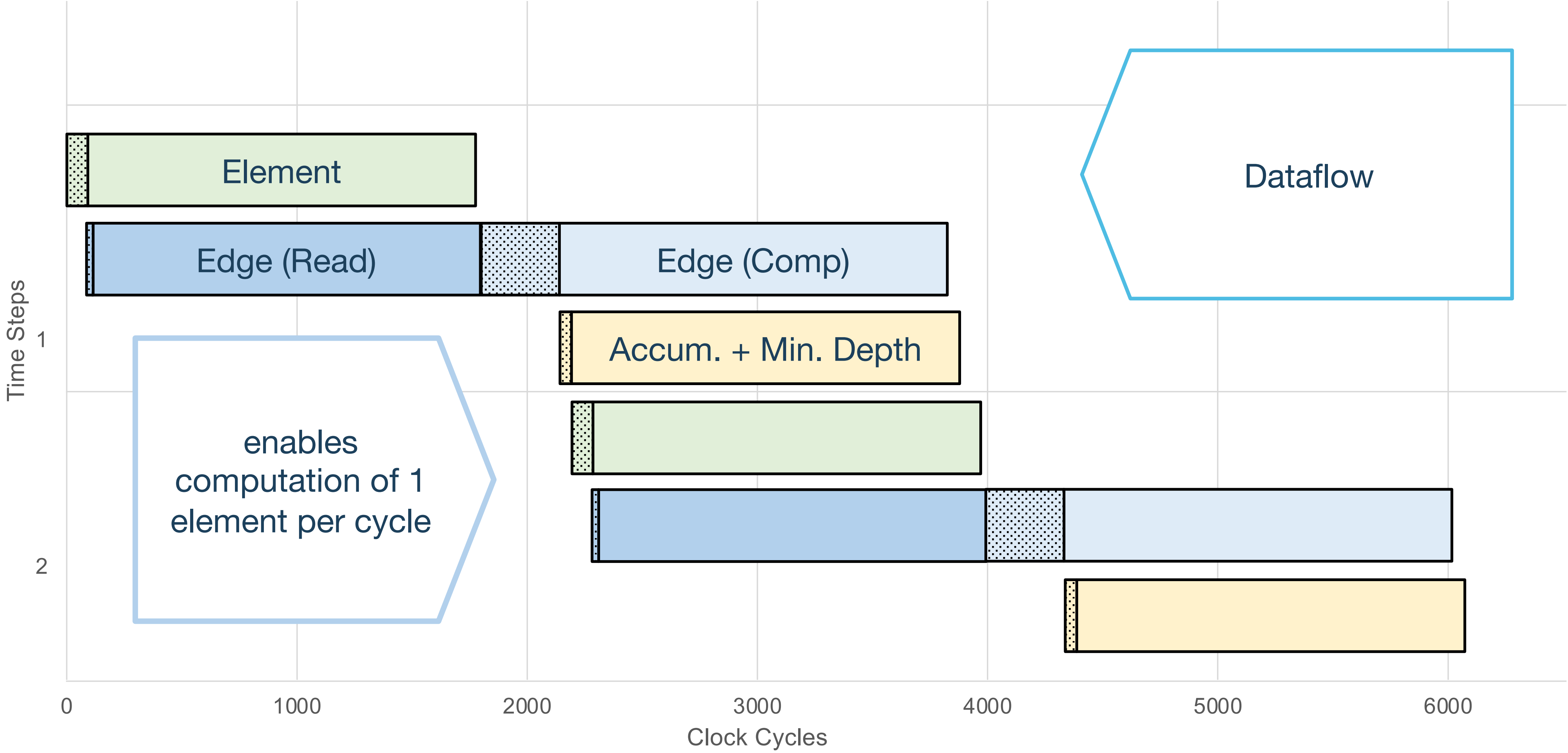


UTBEST2D Single FPGA Design [1]

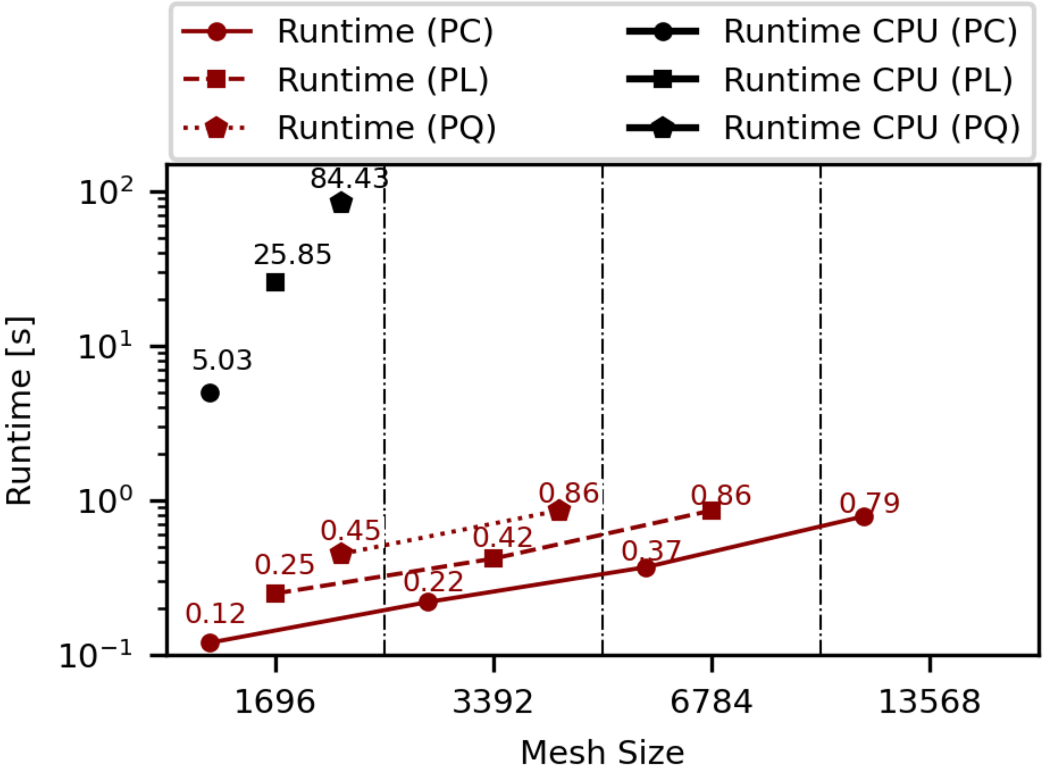


buffer 3 private copies of L^2 projections

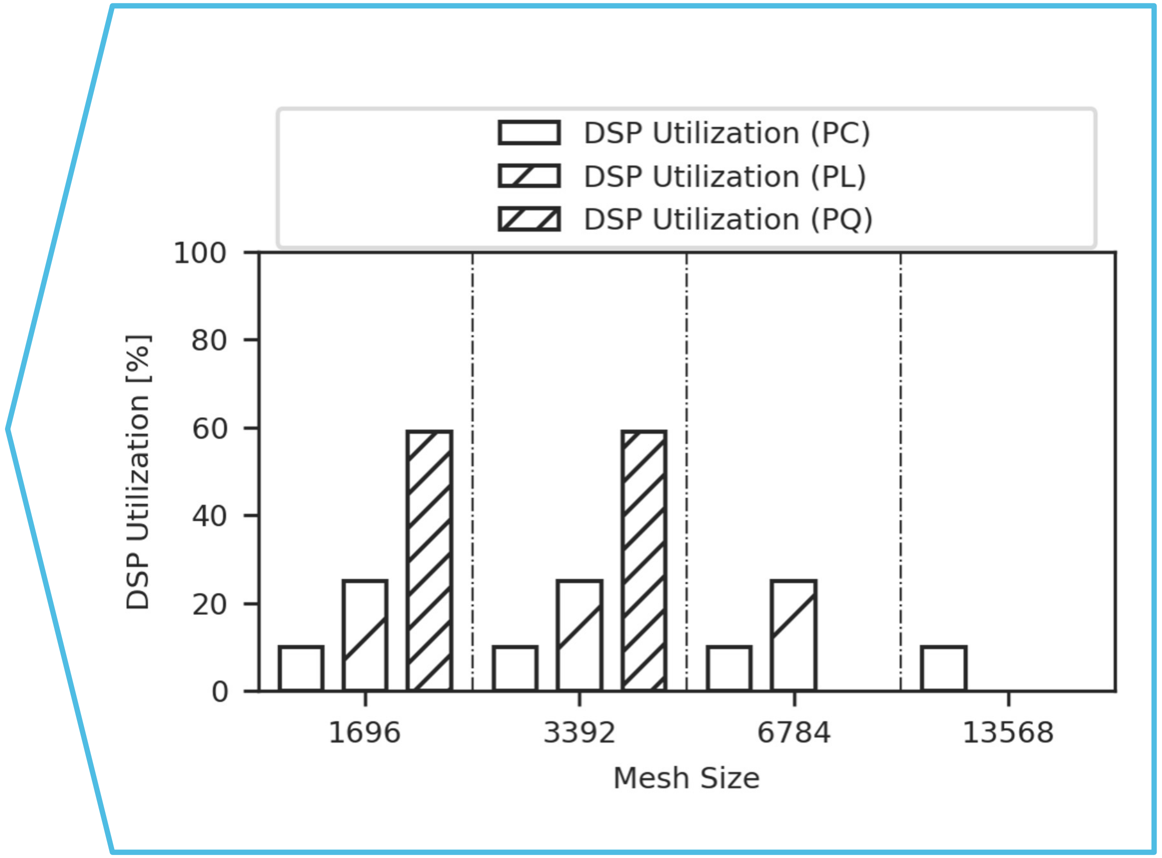
UTBEST2D Single FPGA Design [1]



Single FPGA Results

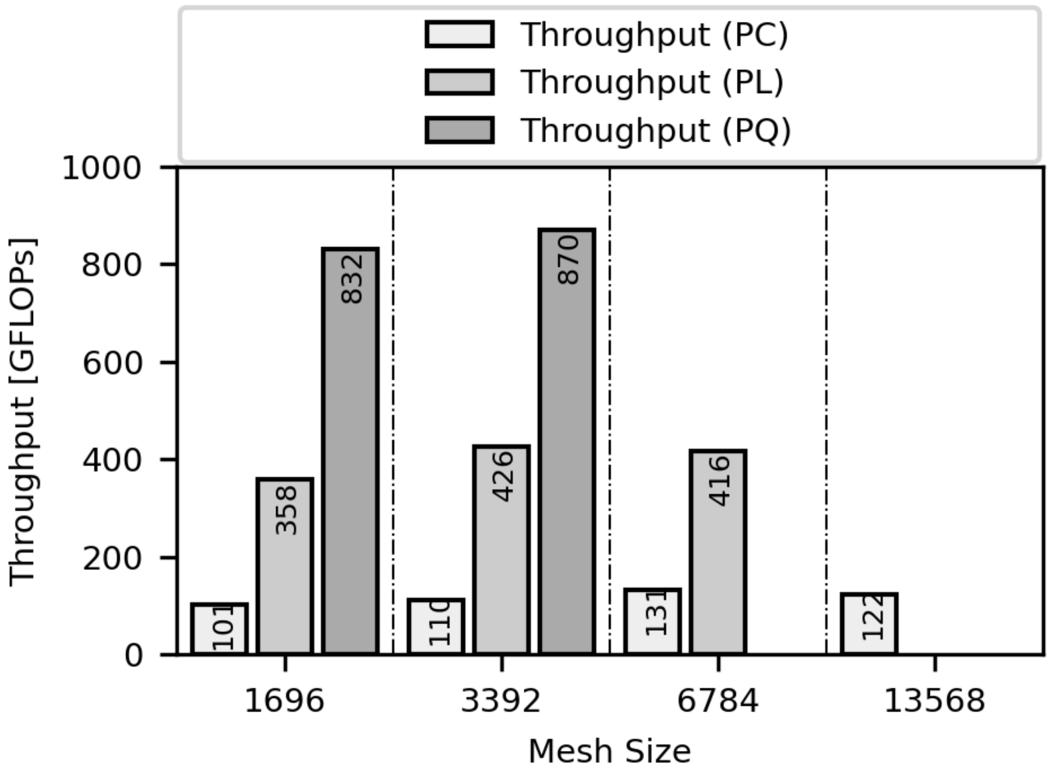


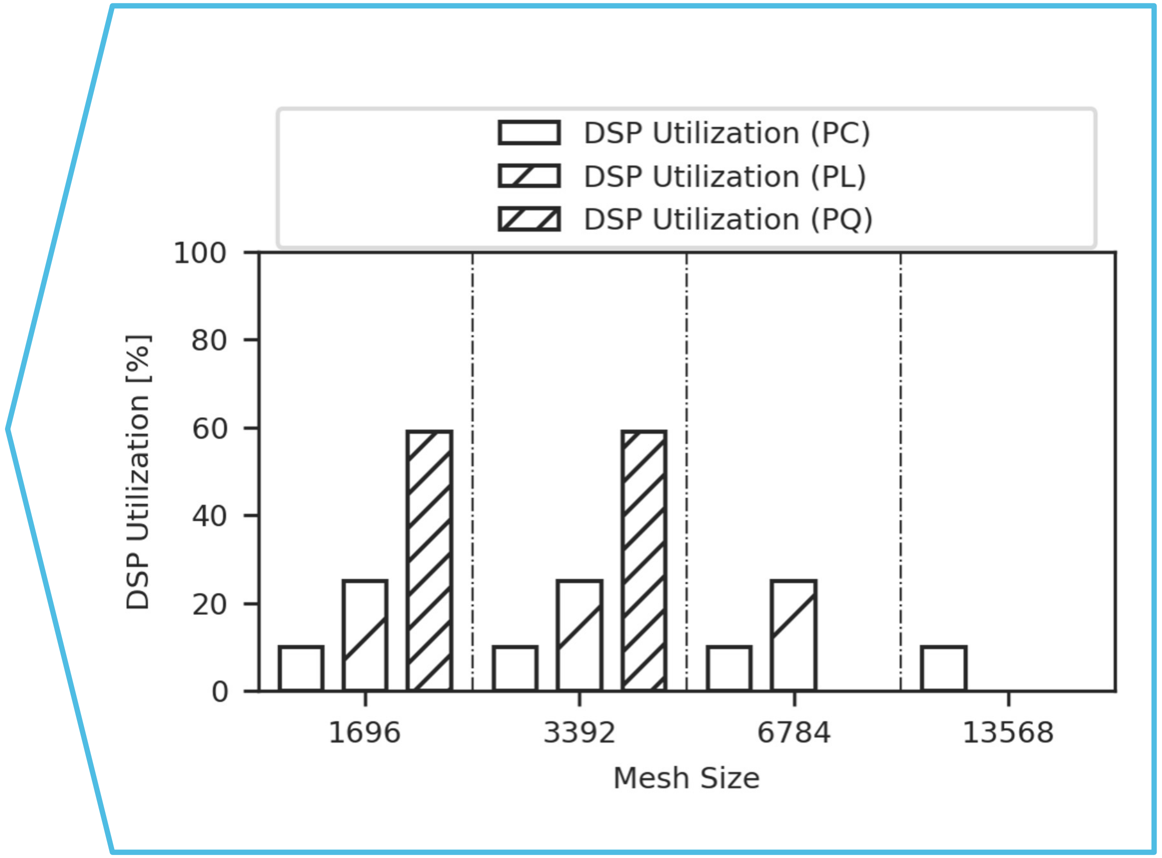
- Stratix 10 GX 2800 FPGA
  - Intel OpenCL SDK for FPGA 21.4
  - Bittware BSP and Intel Quartus Pro 20.4
- Single core of Xeon Gold 6148



DSPs for floating point arithmetic

Single FPGA Results

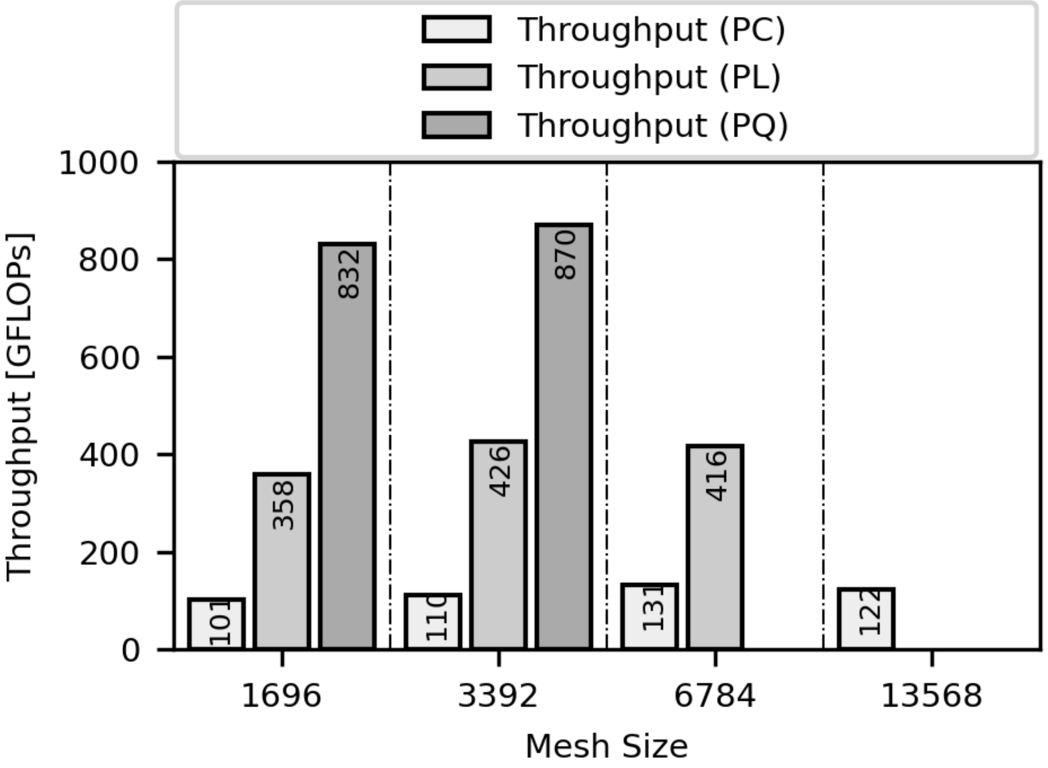


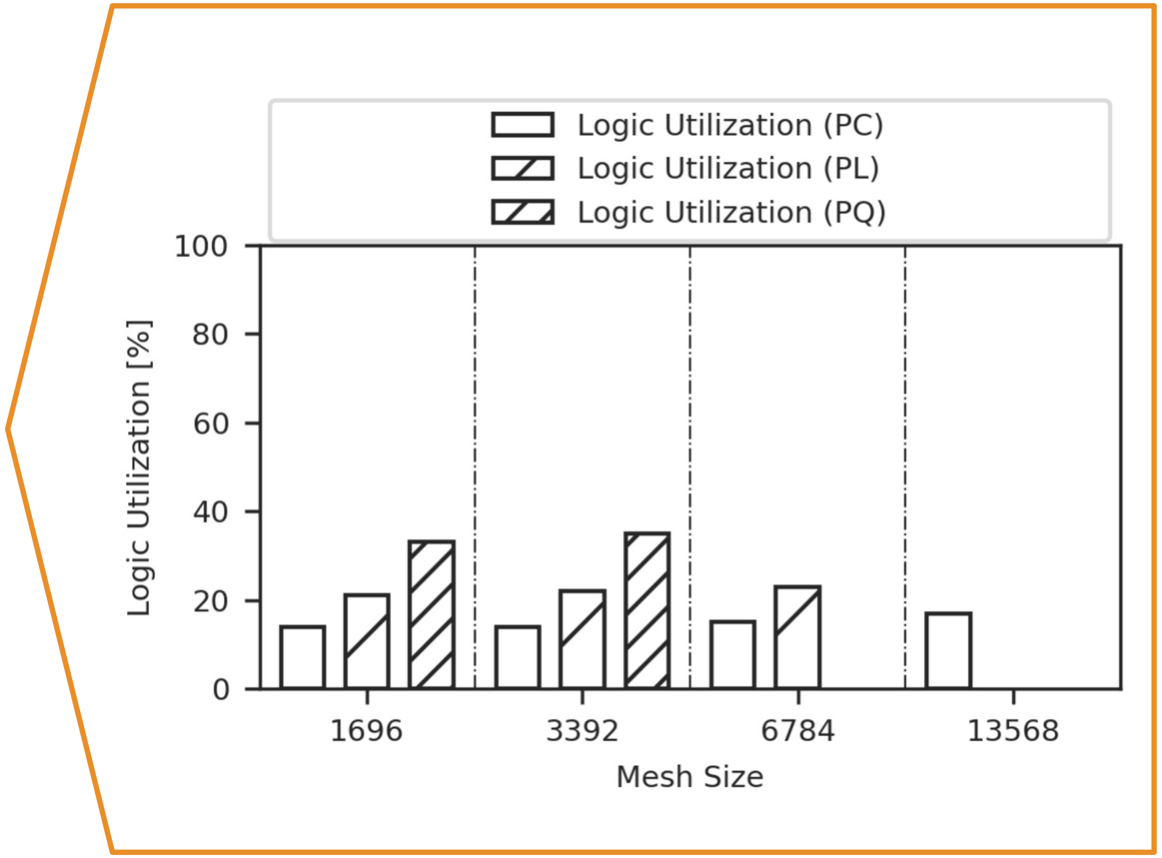


DSPs for floating point arithmetic

FPGA underutilized for PC and PL discretizations

Single FPGA Results

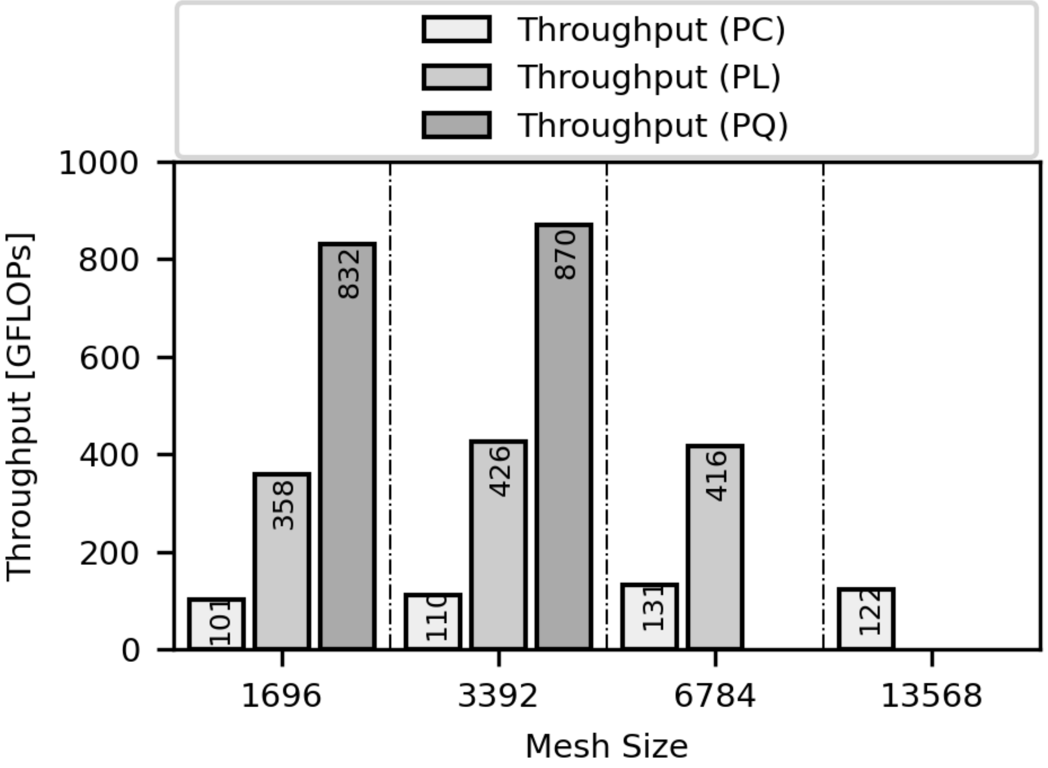


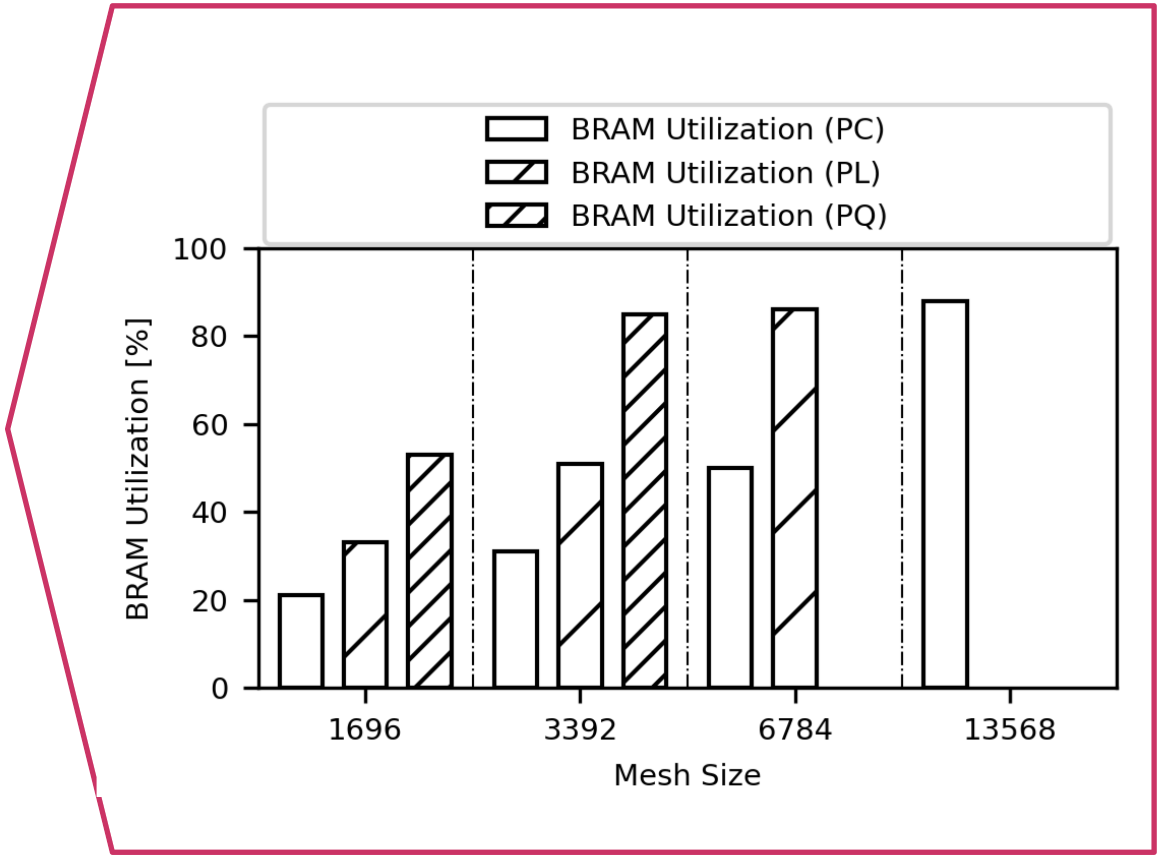


logic for control, index calculations, data movement

FPGA underutilized for PC and PL discretizations

Single FPGA Results



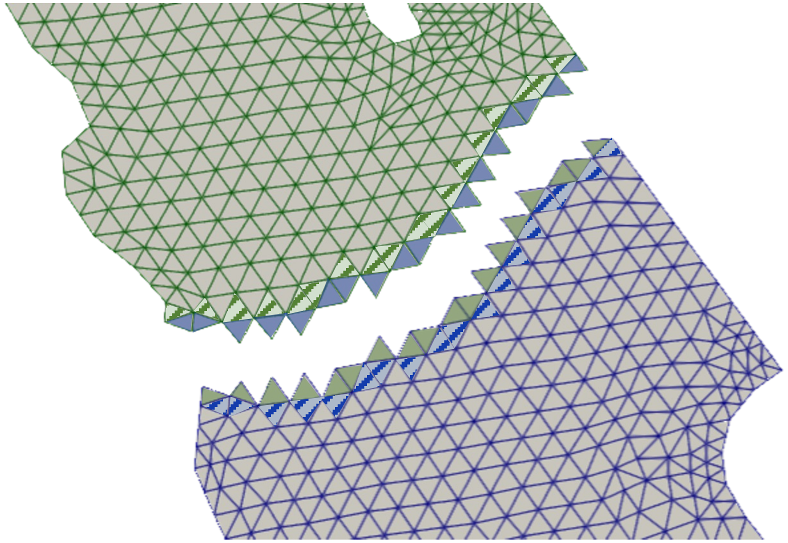


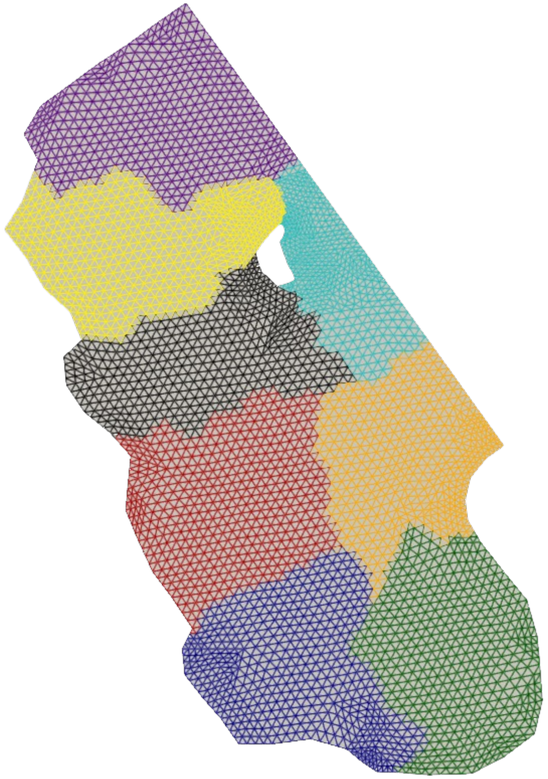
RAM blocks for parallel local memory access

On-chip RAM blocks limit mesh size

Multi-FPGA Parallelism

Partitioned Unstructured Mesh





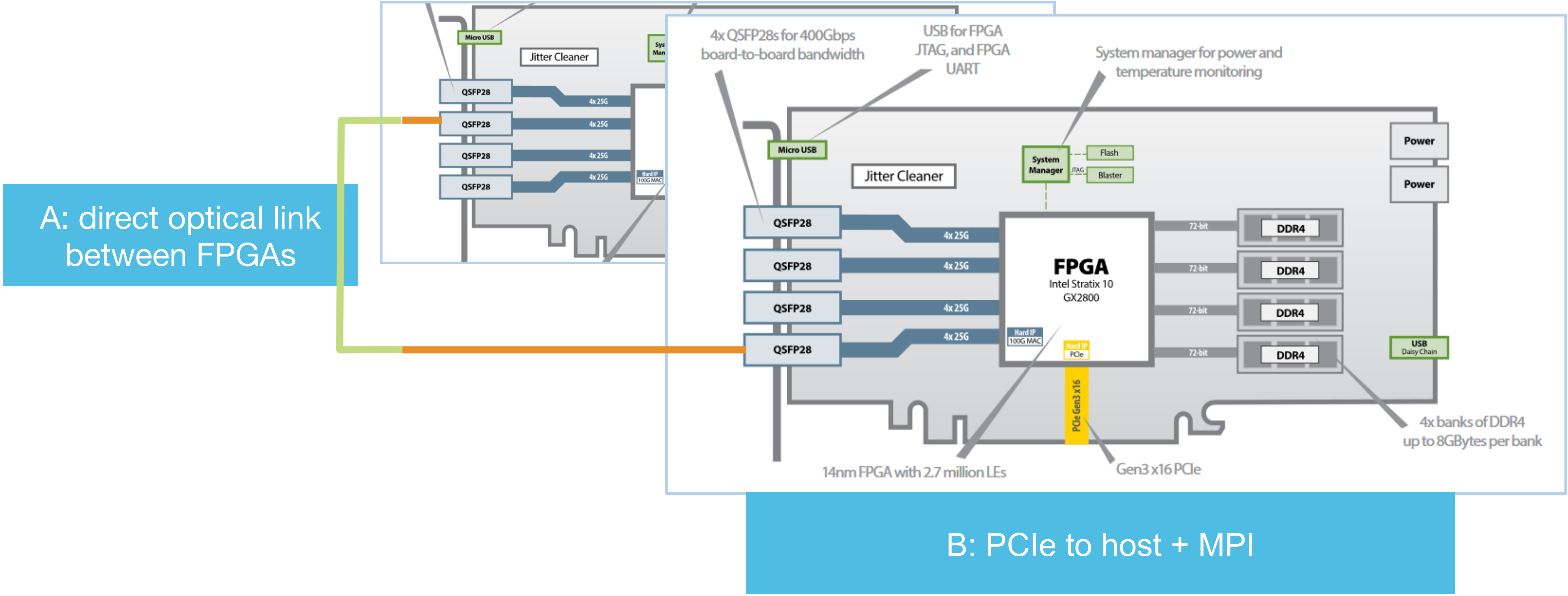
- Spatial partition per device
  - weak scaling with ~ constant memory
  - communicate halo region with neighbors

Excursus:

Inter-FPGA Communication

Communication Options

Bittware 520N with Intel Stratix 10 GX 2800 FPGA [2]



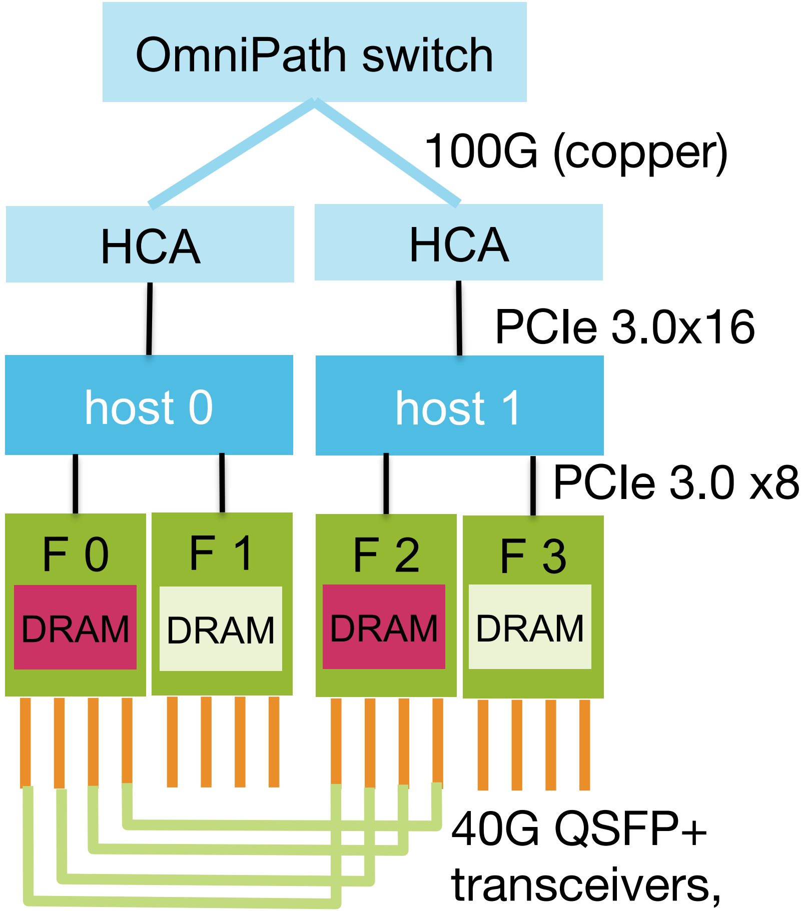
A: direct optical link between FPGAs

B: PCIe to host + MPI

[2] Bittware, 2021.

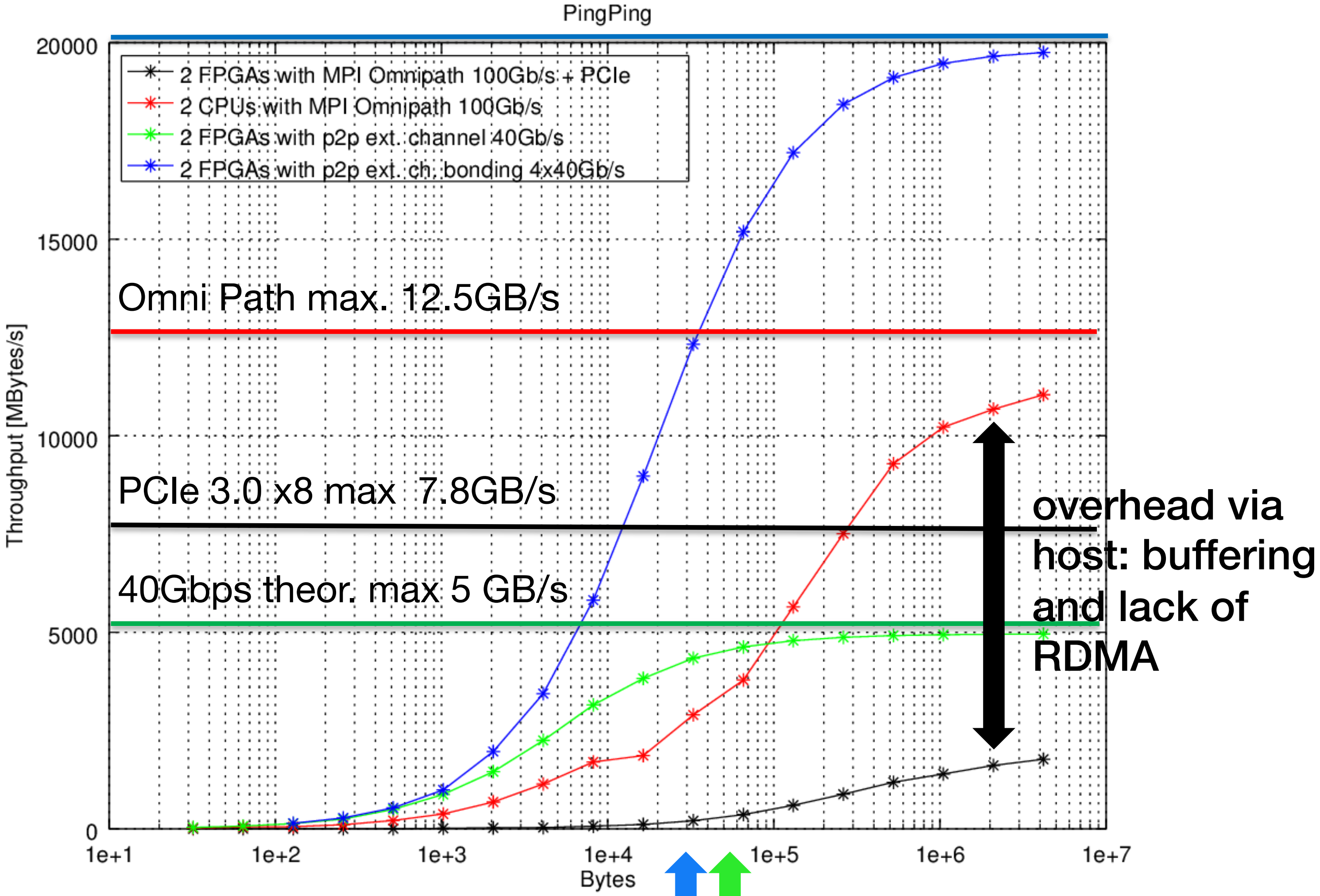
The Case for Option A (direct FPGA communication)

Transfer data from DRAM on F0 to DRAM on F2



40G QSFP+ transceivers, optical fibers

4x40Gbps theoretical max. 20 GB/s



Omni Path max. 12.5GB/s

PCIe 3.0 x8 max 7.8GB/s

40Gbps theor. max 5 GB/s

overhead via host: buffering and lack of RDMA

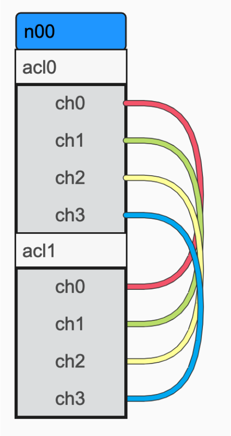
direct FPGA communication faster at lower packet size

* setup changes: Infiniband, cable lengths

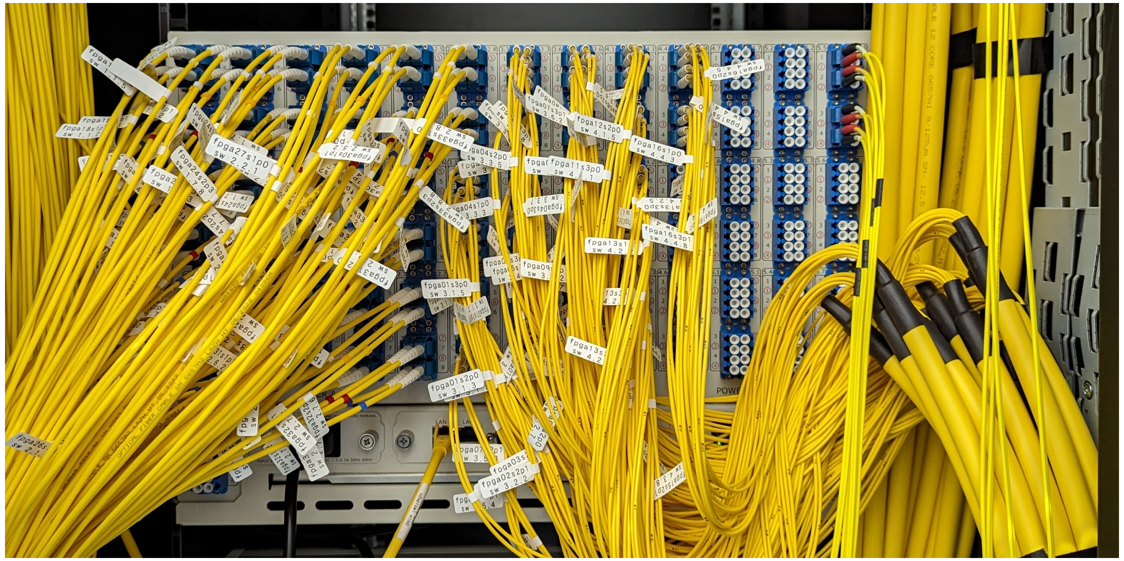
Optical Connections between FPGAs in Noctua 2 HPC System

- 4 direct connections per FPGA
  - configured at job allocation

```
sbatch -N 1 \
  --partition=fpga \
  --constraint=bittware_520n_20.4.0_max \
  --fpgalink="n00:acl0:ch0-n00:acl1:ch0" \
  --fpgalink="n00:acl0:ch1-n00:acl1:ch1" \
  --fpgalink="n00:acl0:ch2-n00:acl1:ch2" \
  --fpgalink="n00:acl0:ch3-n00:acl1:ch3" \
  --t 2:00:00 ./utbest_fpga
```



  - optical circuit switch
    - protocol agnostic
  - streaming protocol on FPGAs



Calient S320

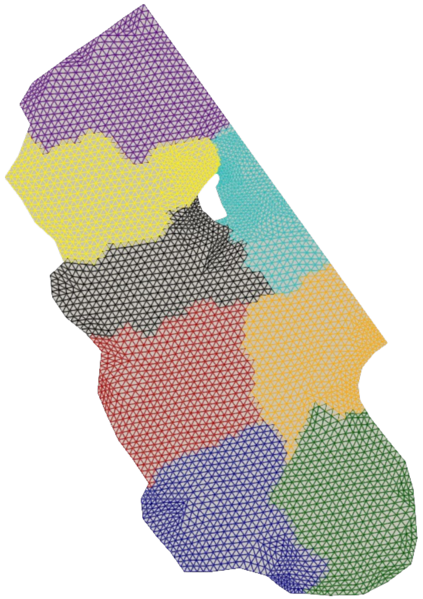
Excursus End

Partitioning the Bahamas Mesh

Partitioning Tool: Metis

Requirements

1. computation balance over partitions
2. low communication-computation ratio
3. <= 4 neighbors per partition



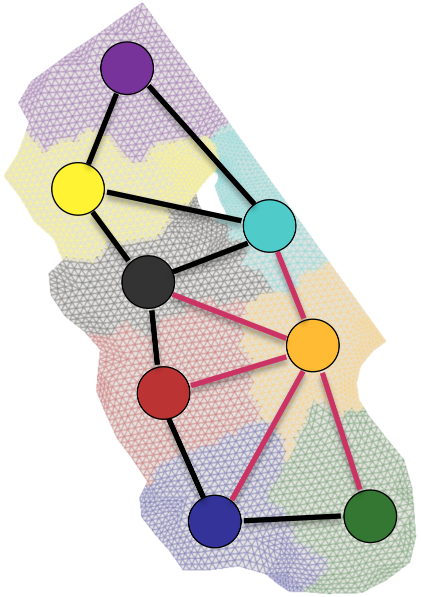
Statistics	E	$E_{boundary}$	Load Imbalance	Comm./Comp.
Min	888	40	0.978	0.05
Max	930	82	1.025	0.09
Avg	908	60	1.000	0.07
Stddev	*14*	*14*	*0.015*	*0.02*

Partitioning the Bahamas Mesh

Partitioning Tool: Metis

Requirements

1. computation balance over partitions
2. low communication-computation ratio
3. <= 4 neighbors per partition



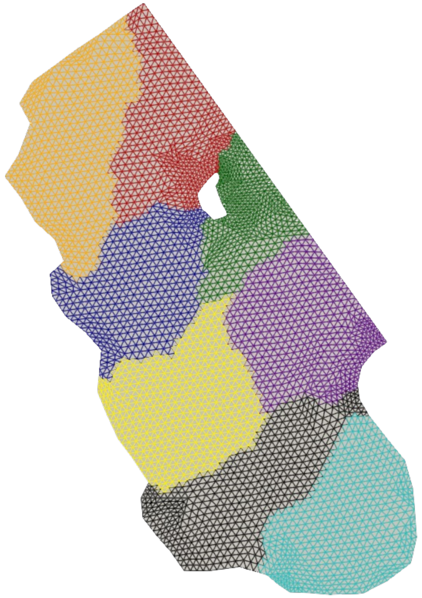
Statistics	E	$E_{boundary}$	Load Imbalance	Comm./Comp.
Min	888	40	0.978	0.05
Max	930	82	1.025	0.09
Avg	908	60	1.000	0.07
Stddev	*14*	*14*	*0.015*	*0.02*

Partitioning the Bahamas Mesh

Partitioning Tool: Metis

Requirements

1. computation balance over partitions
2. low communication-computation ratio
3. <= 4 neighbors per partition



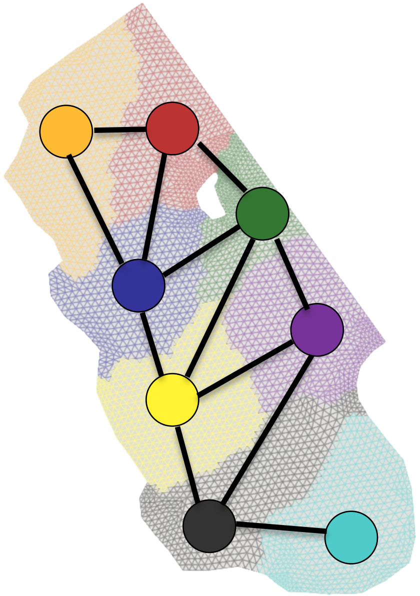
Statistics	E	$E_{boundary}$	Load Imbalance	Comm./Comp.
Min	878	30	0.970	0.03
Max	924	76	1.020	0.08
Avg	906	58	1.000	0.06
Stddev	*15*	*15*	*0.016*	*0.02*

Partitioning the Bahamas Mesh

Partitioning Tool: Metis

Requirements

1. computation balance over partitions
2. low communication-computation ratio
3. <= 4 neighbors per partition

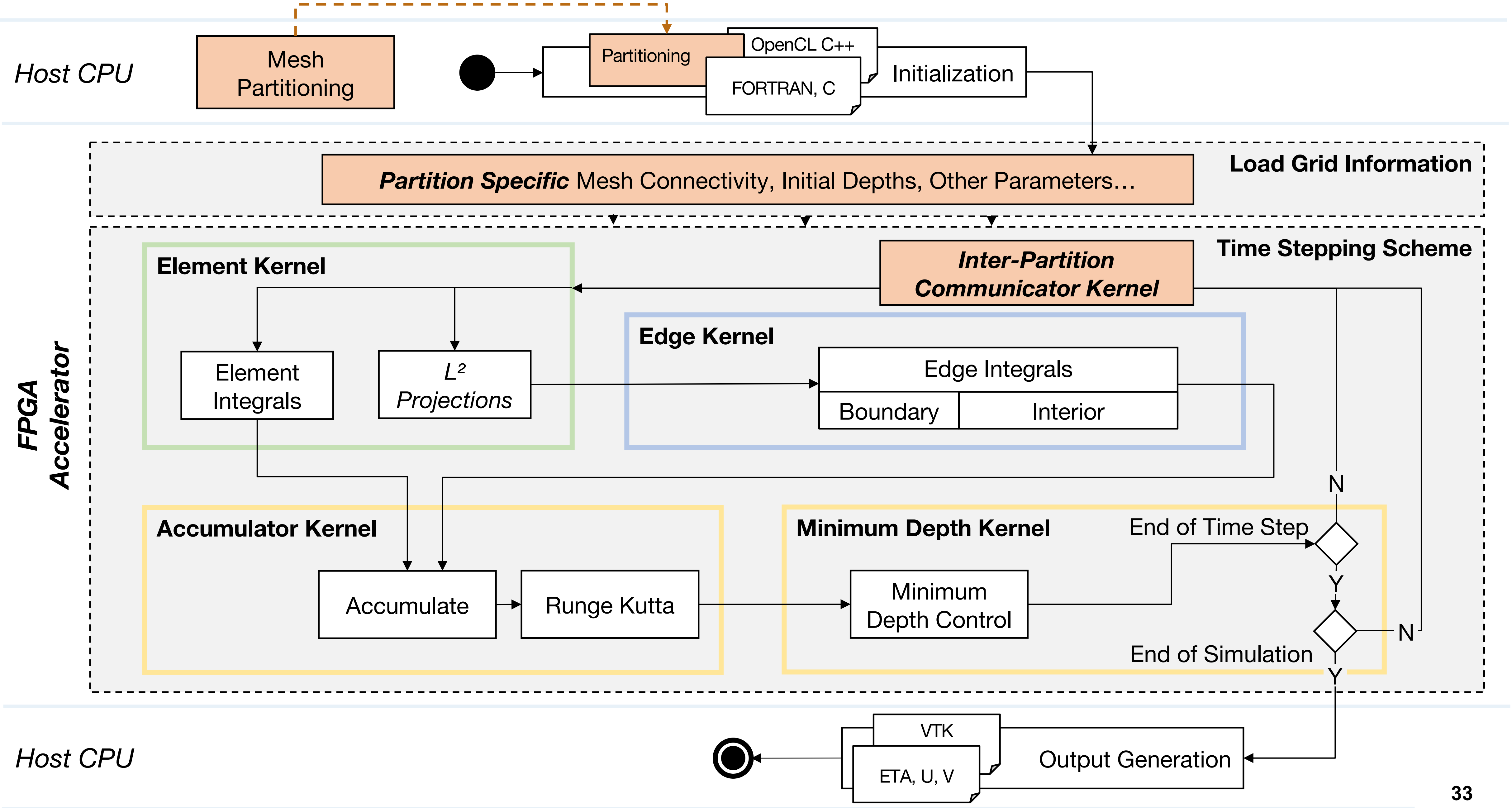


Statistics	E	$E_{boundary}$	Load Imbalance	Comm./Comp.
Min	878	30	0.970	0.03
Max	924	76	1.020	0.08
Avg	906	58	1.000	0.06
Stddev	*15*	*15*	*0.016*	*0.02*

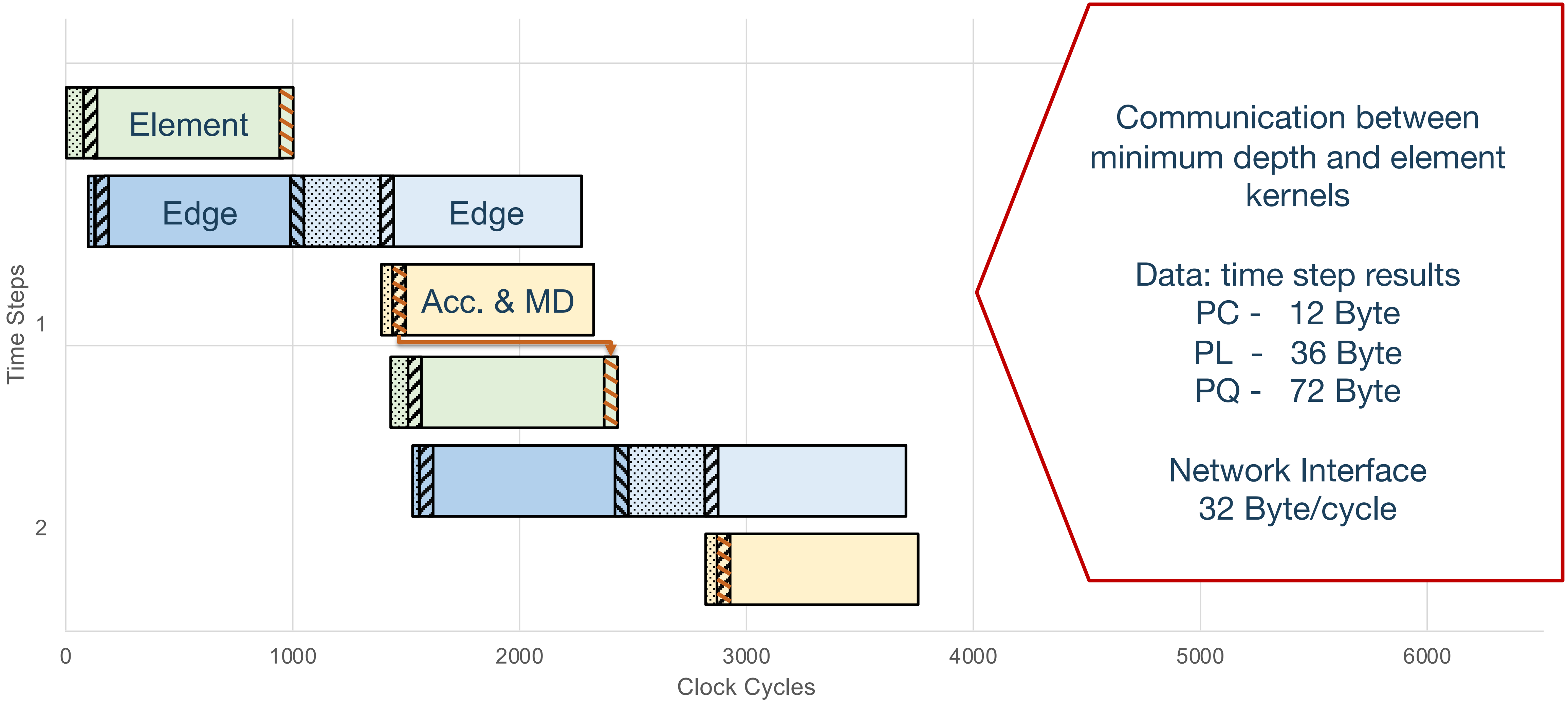
Multi FPGA Design

(Remotely Partitioned)

Extension: Partitioned Execution



Execution Schedule (per partition)

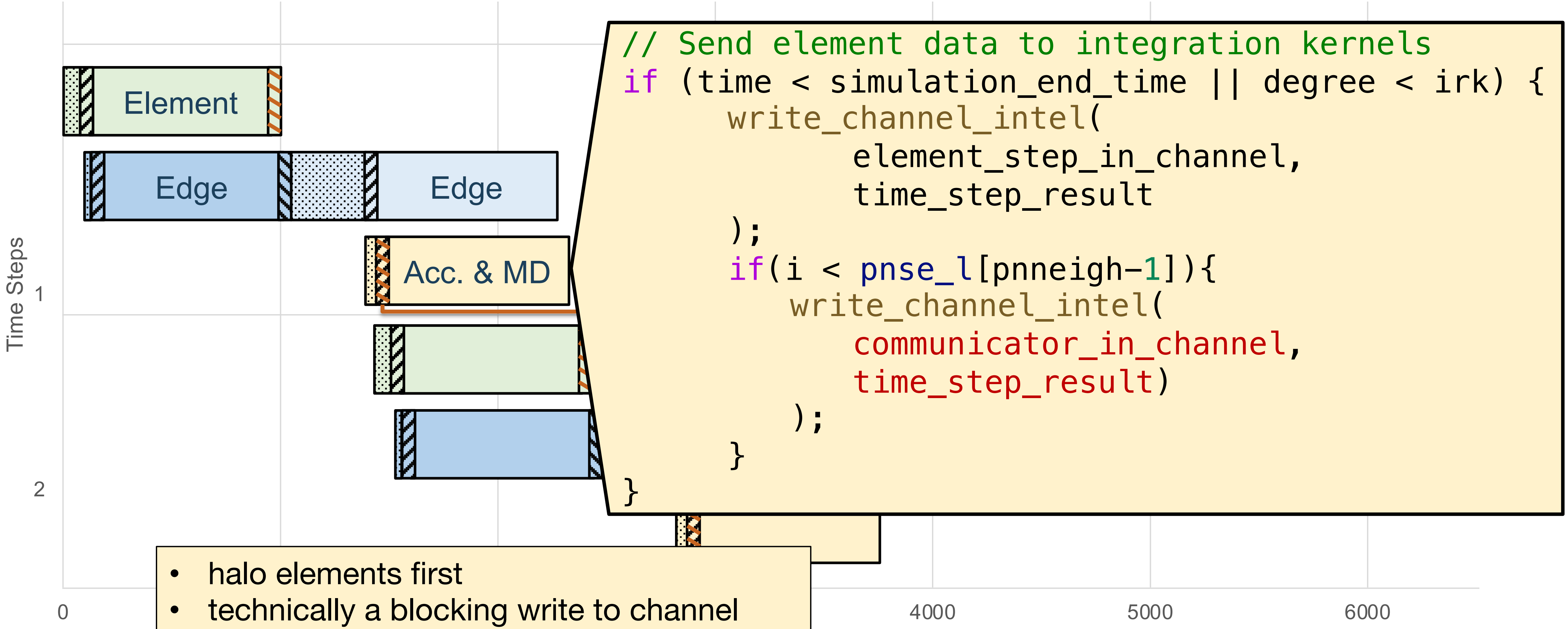


Communication between minimum depth and element kernels

Data: time step results
PC - 12 Byte
PL - 36 Byte
PQ - 72 Byte

Network Interface
32 Byte/cycle

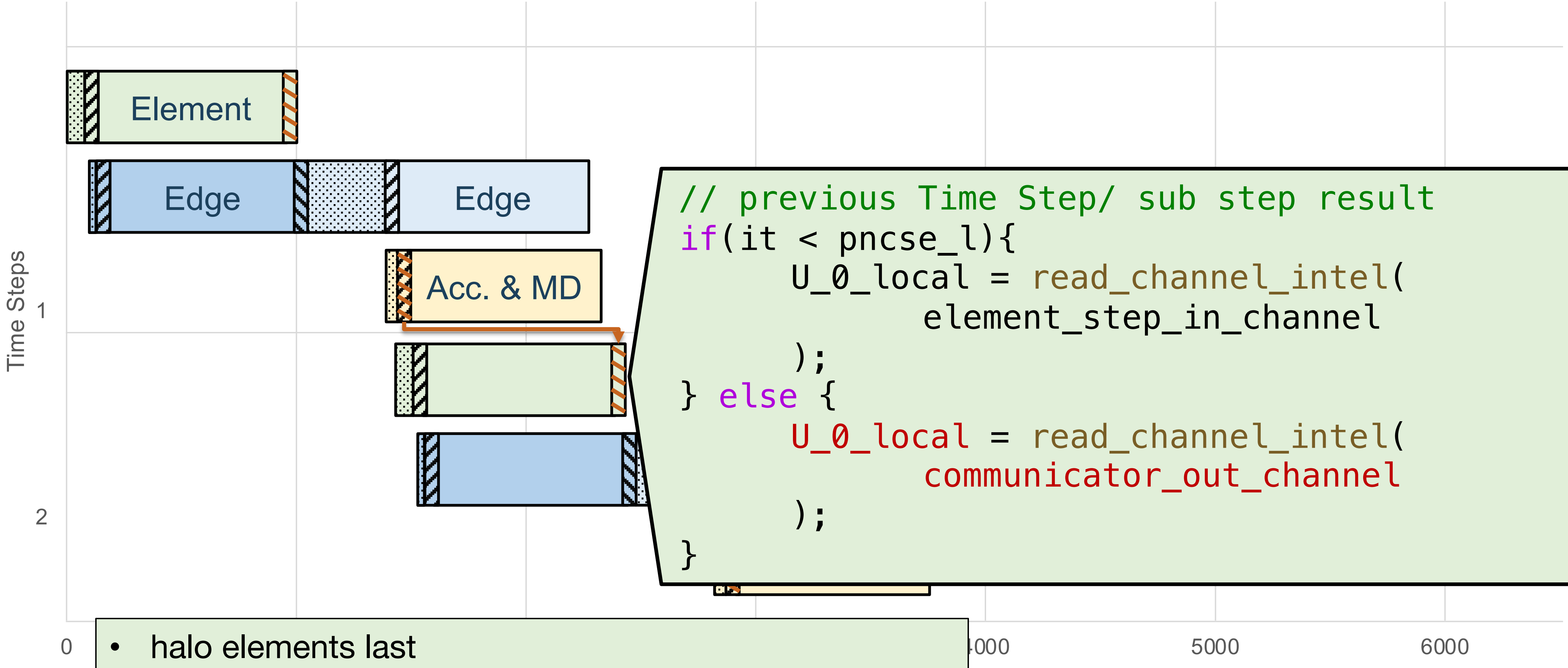
Execution Schedule (sending halos)



```
// Send element data to integration kernels
if (time < simulation_end_time || degree < irk) {
        write_channel_intel(
                element_step_in_channel,
                time_step_result
        );
        if(i < pnse_l[pnneigh-1]){
            write_channel_intel(
                communicator_in_channel,
                time_step_result)
            );
        }
}
```

- halo elements first
- technically a blocking write to channel
- but channel has a capacity (depth)
- effectively performs as non-blocking send

Execution Schedule (receiving halos)

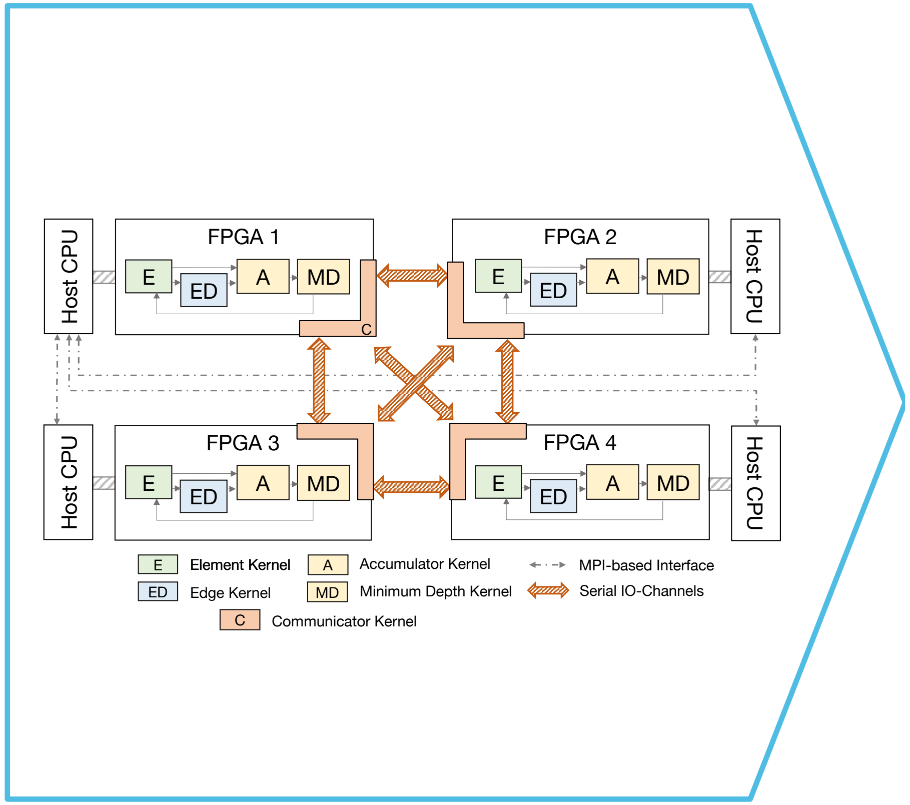


```
// previous Time Step/ sub step result
if(it < pncse_l){
        U_0_local = read_channel_intel(
                 element_step_in_channel
        );
} else {
        U_0_local = read_channel_intel(
                 communicator_out_channel
        );
}
```

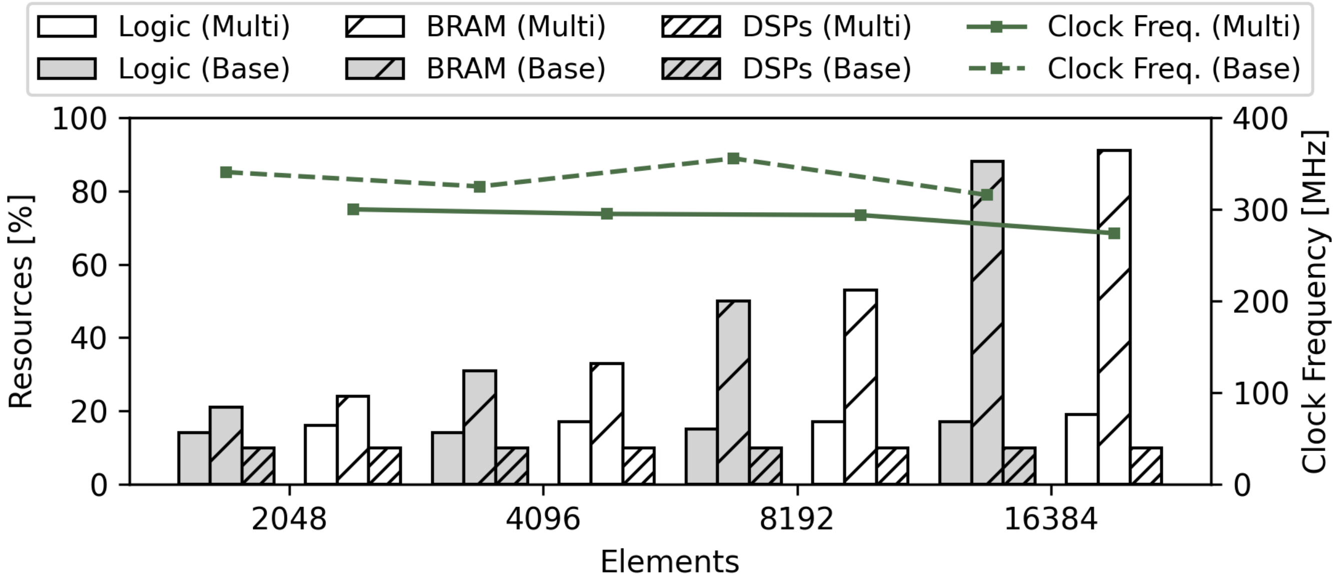
- halo elements last
- blocking read from channel
- latency covered by computations on internal elements

Synthesized Designs
+ Evaluation

Multi-FPGA Design (Remotely Partitioned)



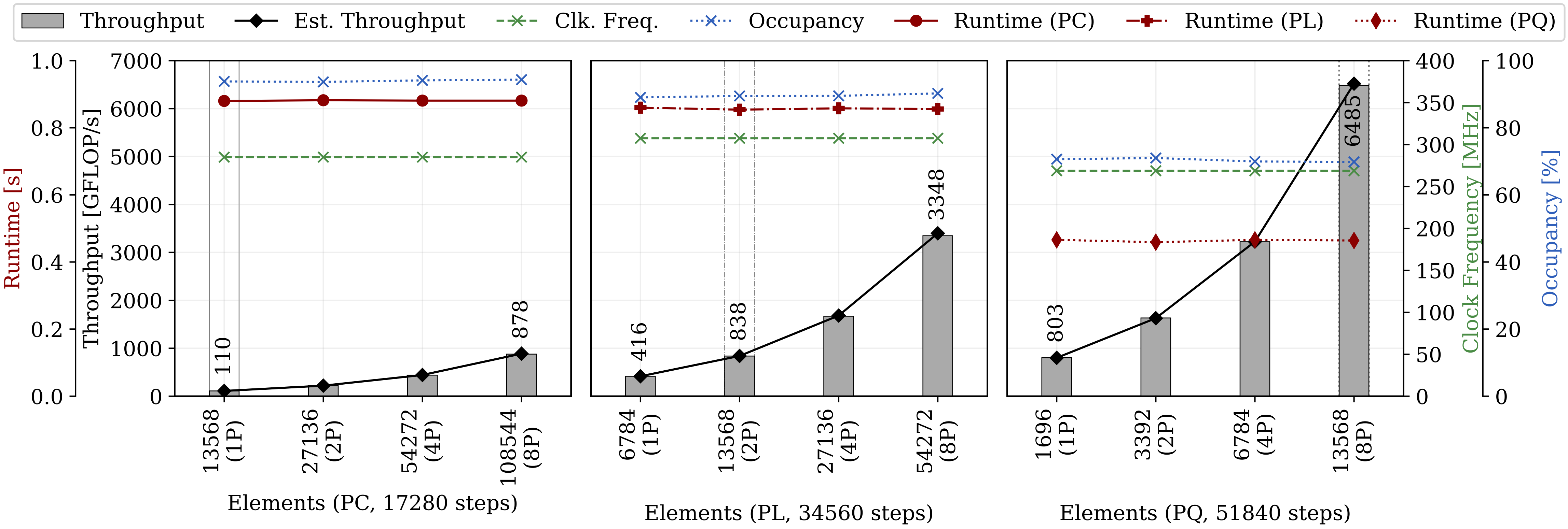
Piecewise Constant (PC) Design



Suitable for PC, PL, PQ

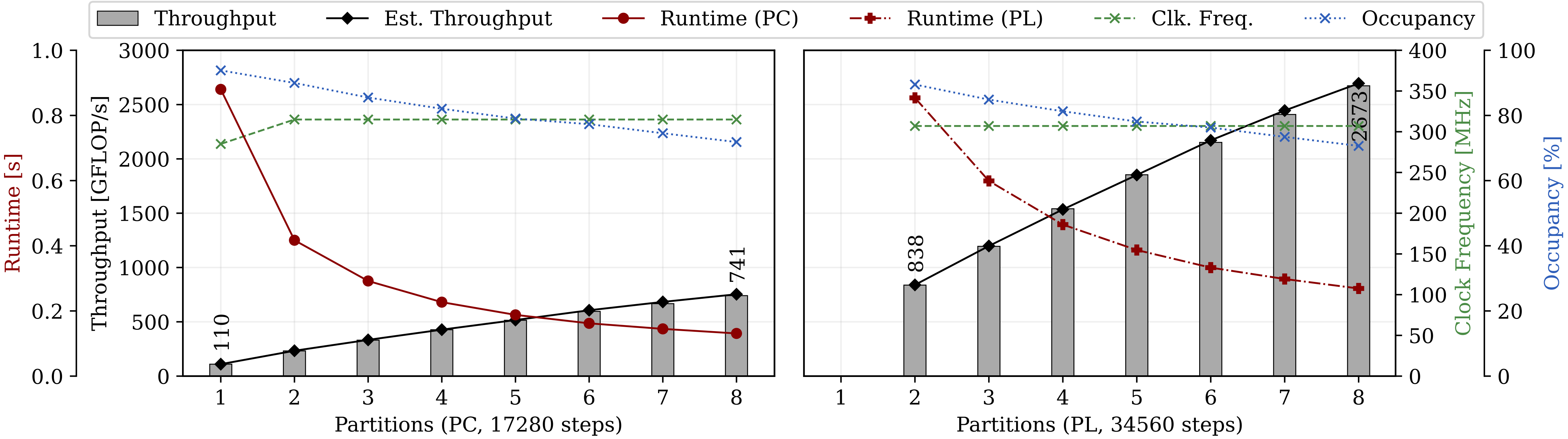
Can run with as many partitions as connected FPGAs are available

Multi-FPGA Design – Weak Scaling



- accurate performance model with occupancy (modeled) and clock frequency
- combined local memory capacity of multiple FPGAs for larger problem sizes

Multi-FPGA Design – Strong Scaling

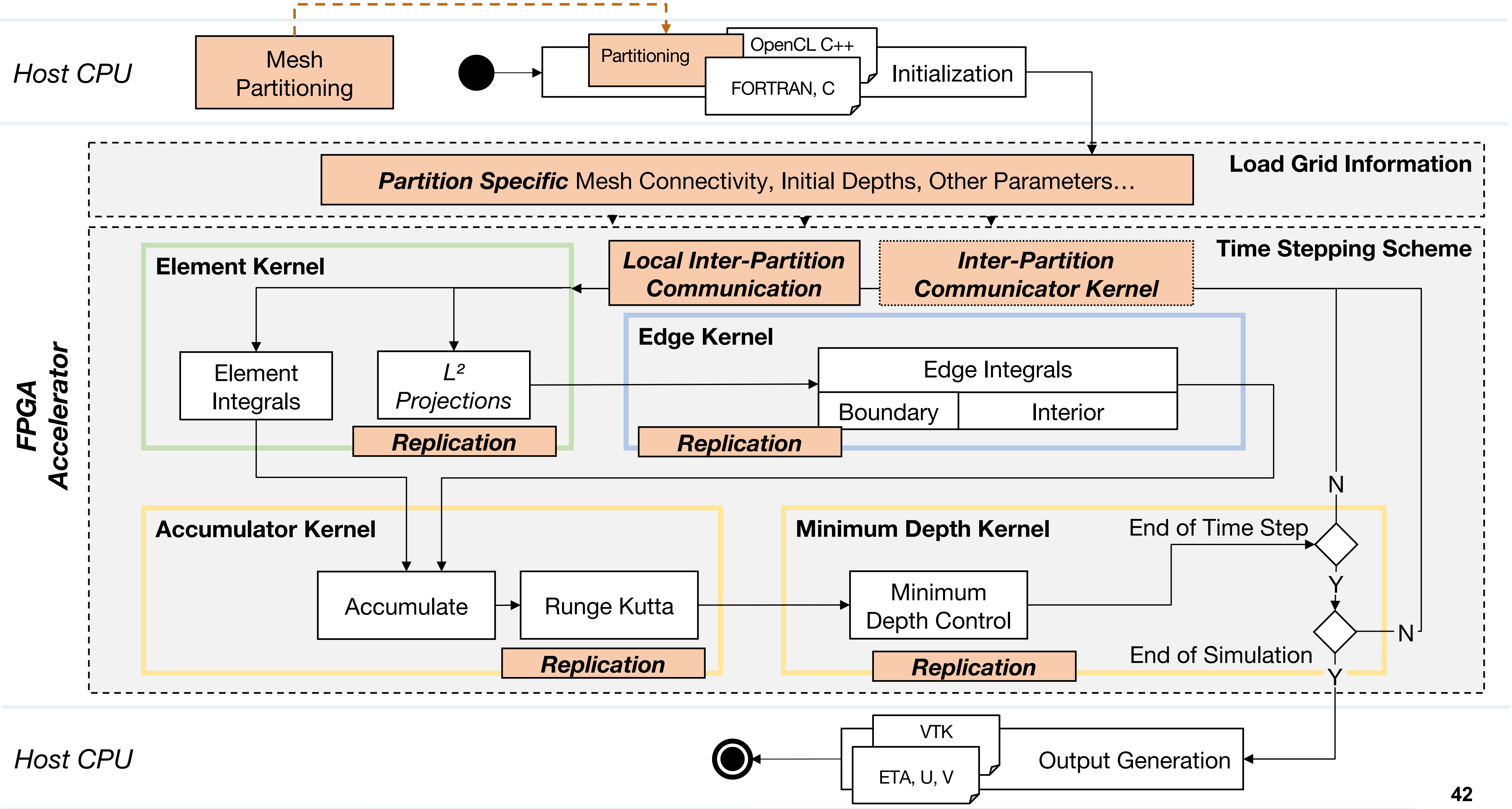


- 13568 elements distributed to N partitions
  - occupancy decreasing due to constant pipeline latency contribution
  - no effect of communication latency here

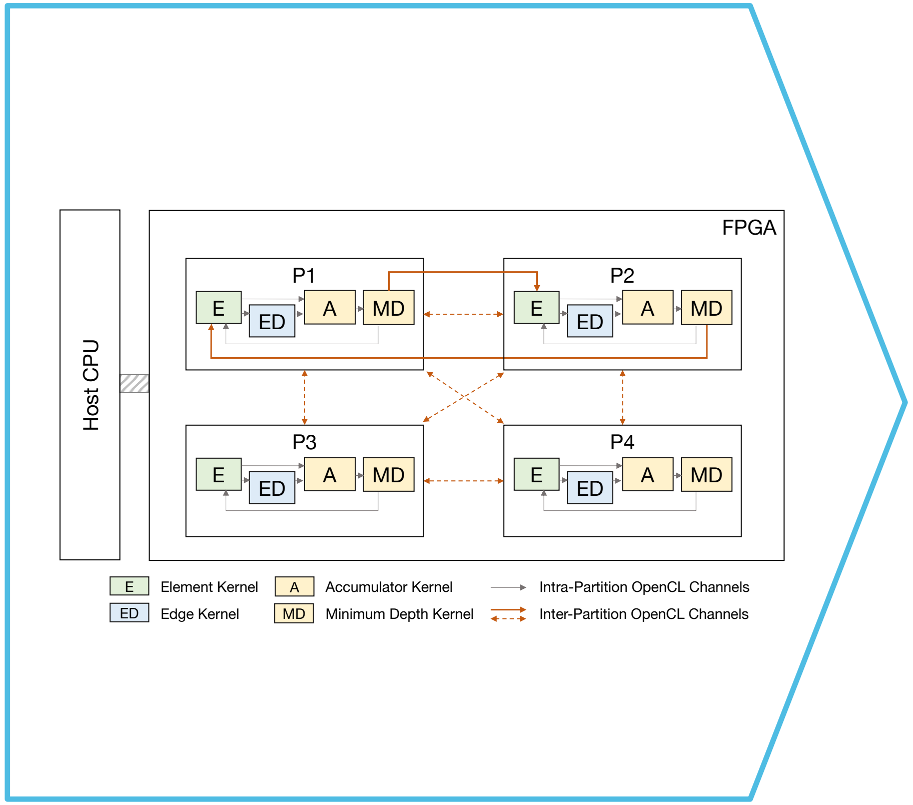
Multi Partition Design

(Locally Partitioned, Hierarchically Partitioned)

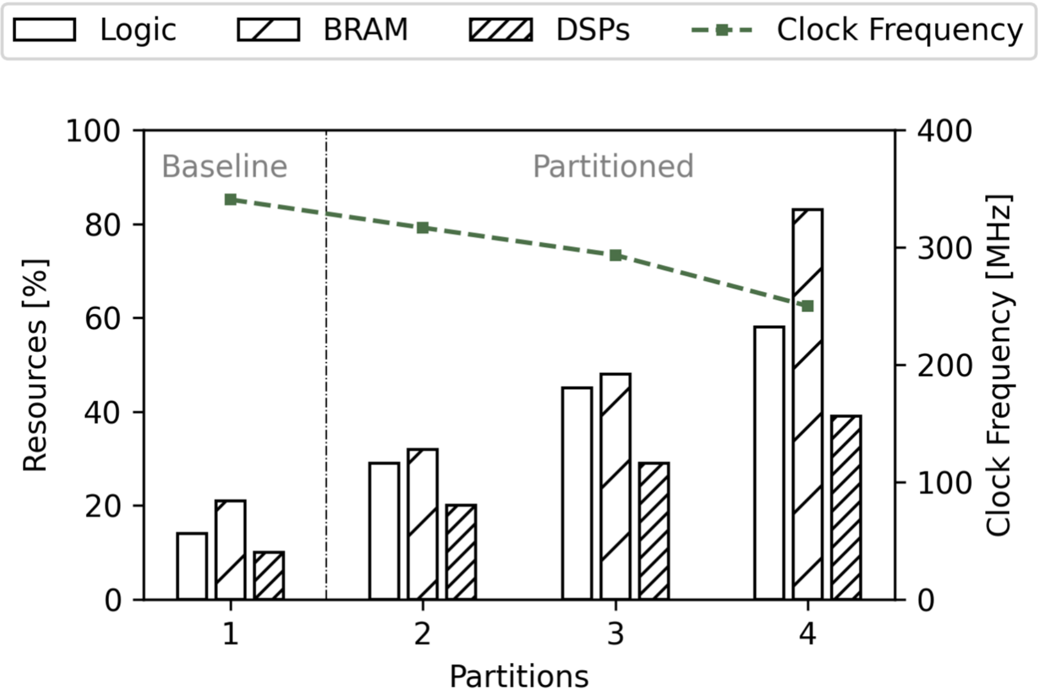
Extension: Partitioned Execution per FPGA



Single FPGA Partitioned Design (Locally Partitioned)

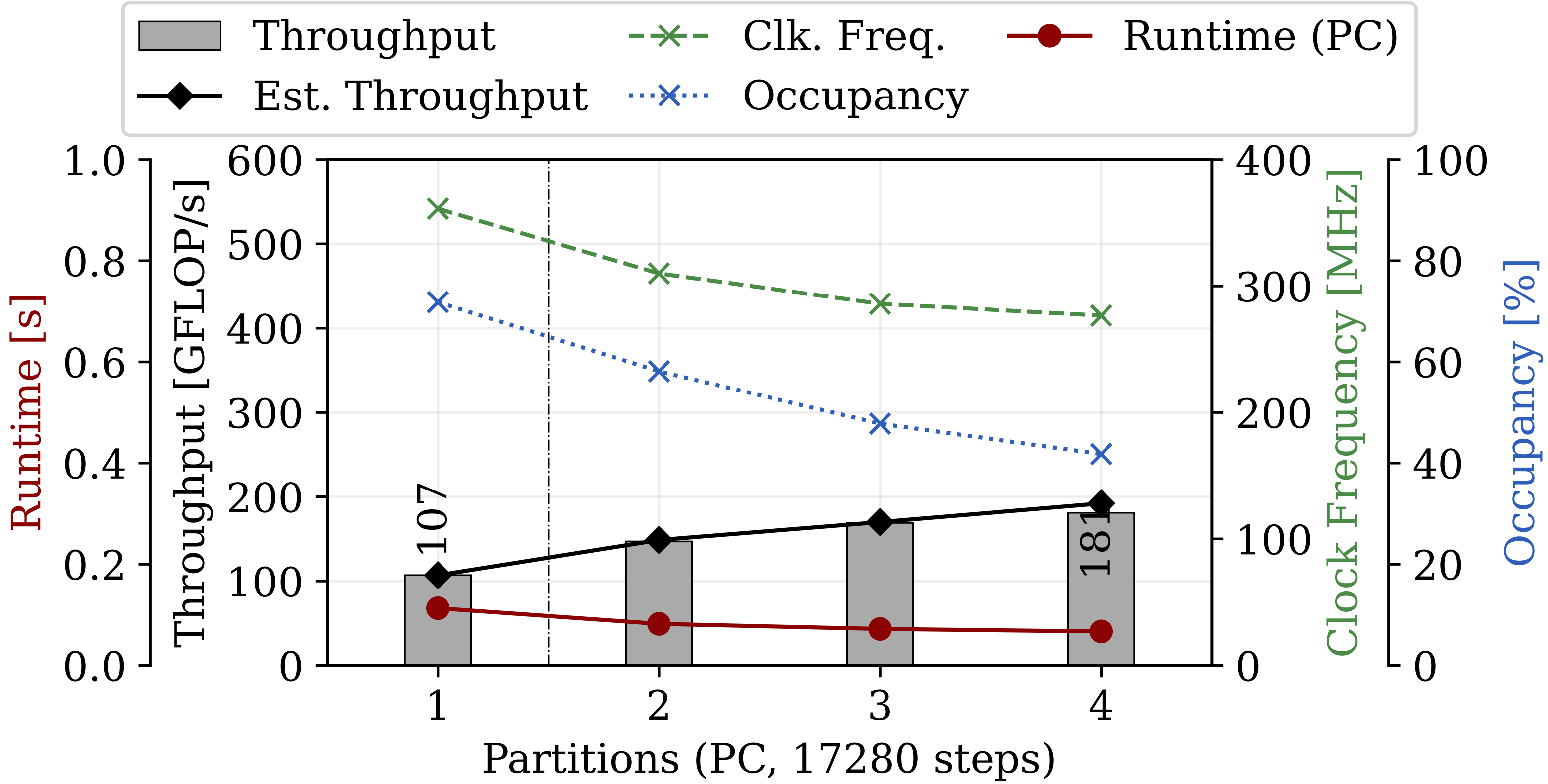


Piecewise Constant (PC) Design



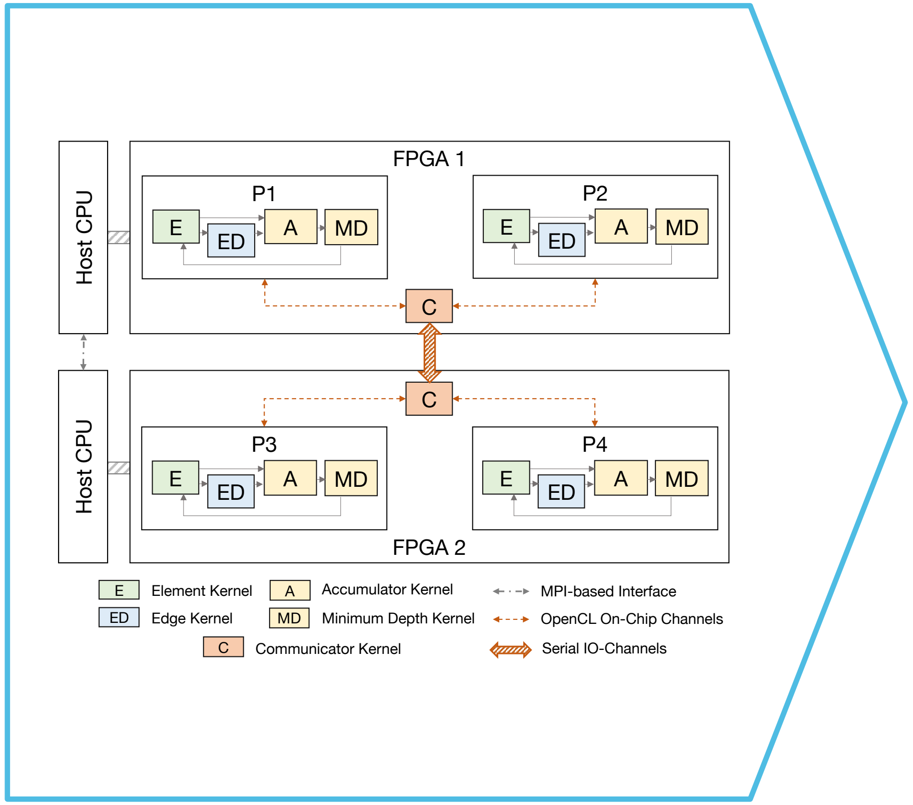
Suitable for PC, PL (in underutilized resources)
Can run with as many partitions as fit on single FPGA

Single FPGA Partitioned Design – Strong Scaling

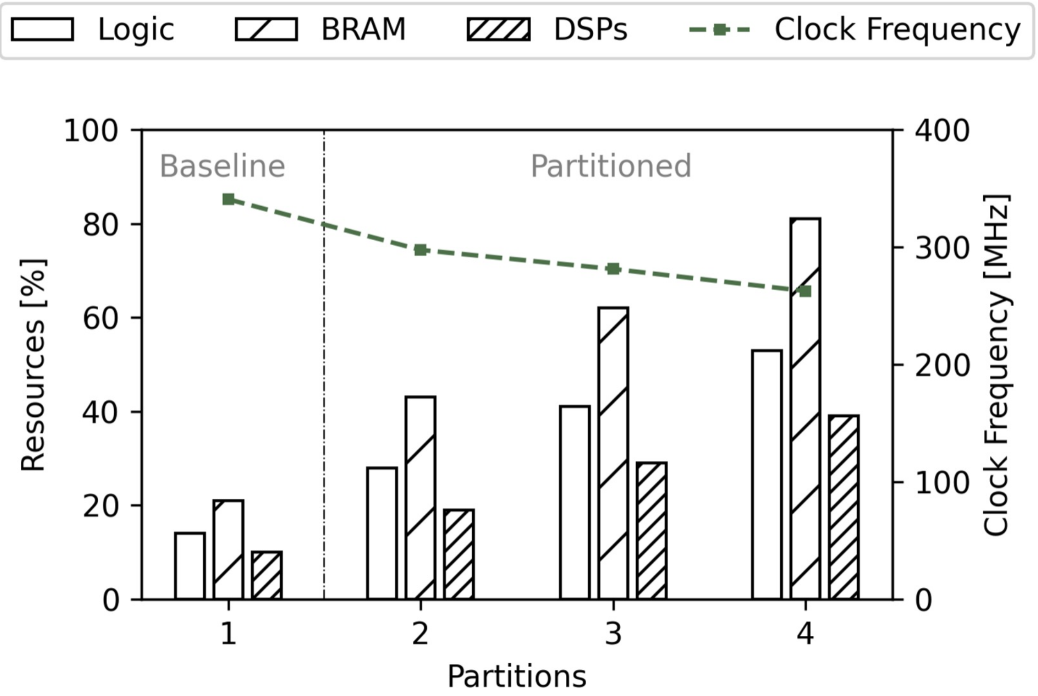


- Local partitions compete for local memory resources
  - here: strong scaling with 1696 elements
- Combined effects of clock frequency and pipeline latency limit speedup to ~1.7x

Hierarchically Partitioned Design



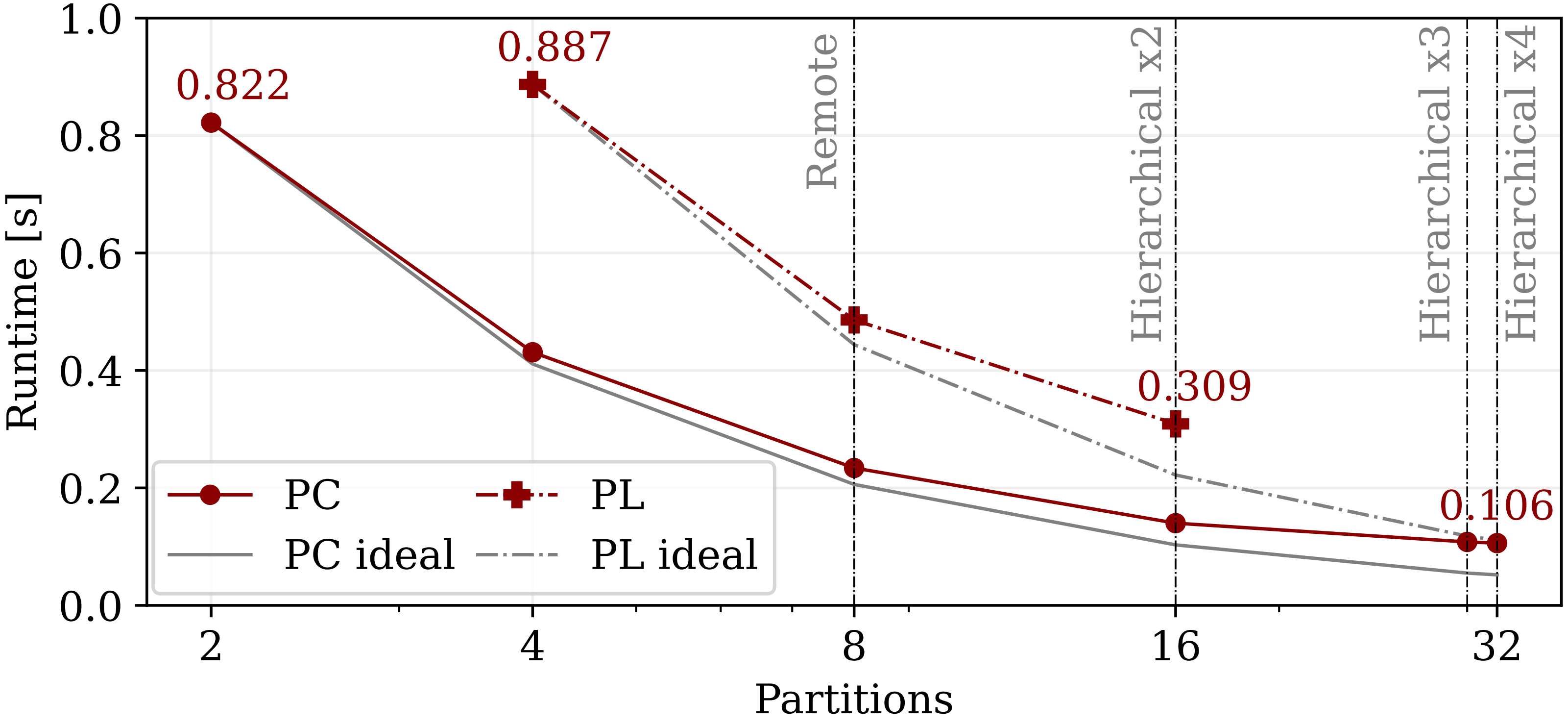
Piecewise Constant (PC) Design

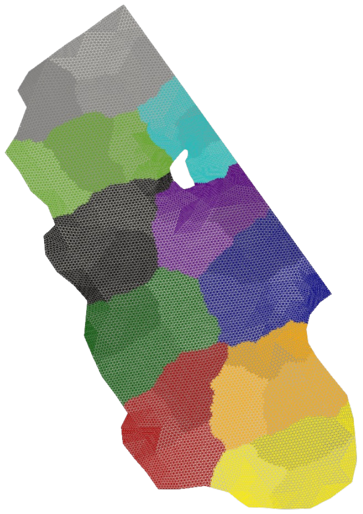


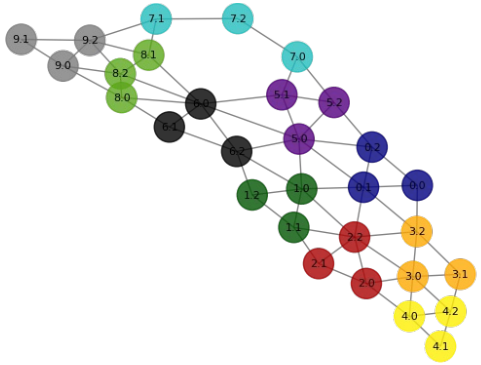
Suitable for PC, PL (in underutilized resources)

Can run with number of local partitions times number of FPGA

Hierarchically Partitioned Design – Strong Scaling







- 27136 elements distributed to N partitions
  - starting with smallest number of FPGAs required
  - up to 8 FPGAs remotely partitioned only
  - hierarchical design allows further partitions and speedups

Conclusion

Conclusion

- Multi-FPGA scaling for performance and larger problem sizes
- Even promising strong scaling results
- Streaming communication in pipeline allows perfect latency hiding

- Additional benefits from local and hierarchical partitions

Outlook

- Go beyond topologies with fixed connectivity