Towards Scalable and Efficient FPGA Stencil Accelerators

Gaël Deest[1] Nicolas Estibals[1] Tomofumi Yuki[2]
Steven Derrien[1] Sanjay Rajopadhye[3]

[1]IRISA / Université de Rennes 1 / Cairn [2]INRIA / LIP / ENS Lyon
[3]Colorado State University

January 19th, 2016

Stencil Computations

Important class of algorithms

- Iterative grid update.
- Uniform dependences.

Examples:

- Solving partial differential equations
- Computer simulations (physics, seismology, etc.)
- (Realtime) image/video processing

Strong need for efficient hardware implementations.

Application Domains

Two main application types with vastly $\neq$ goals:

HPC

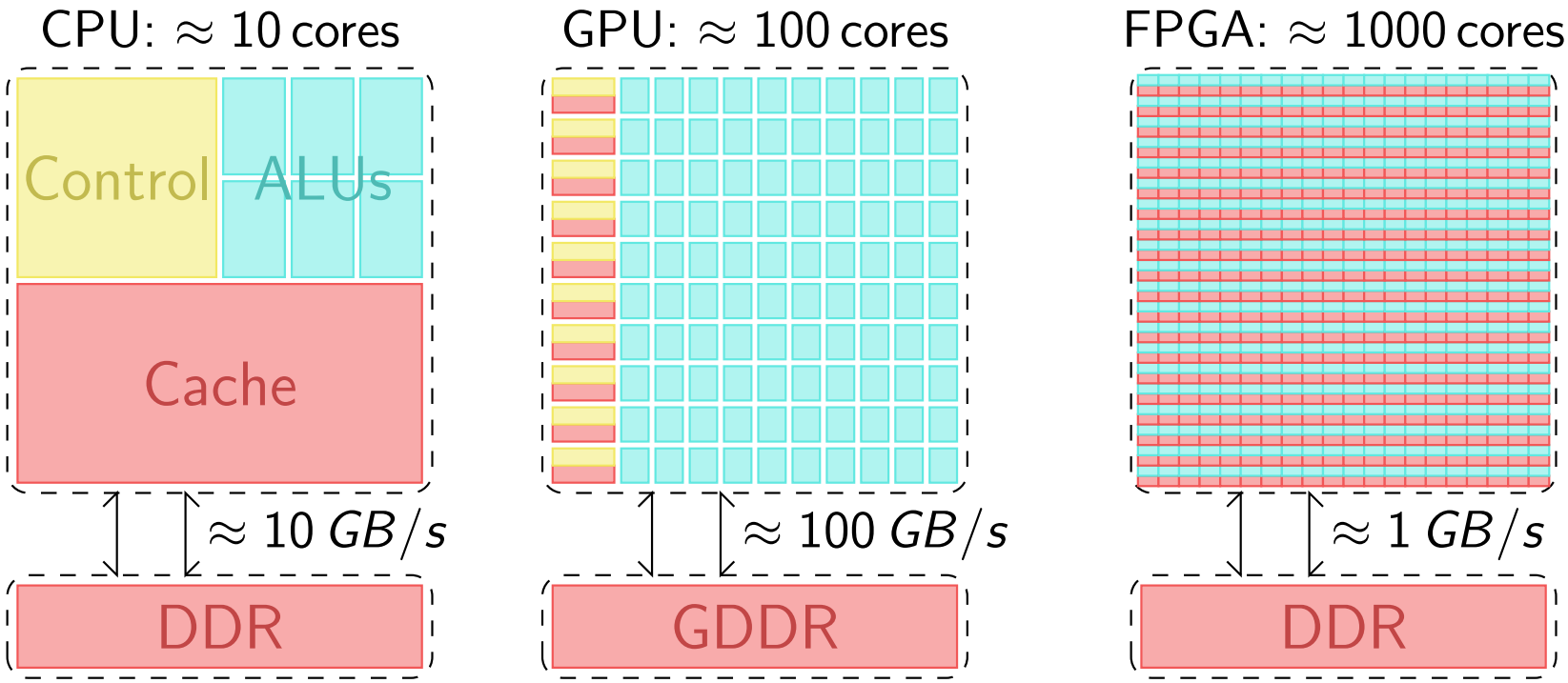
- "Be as fast as possible"
- No realtime constraints

Embedded Systems

- "Be fast enough"
- Realtime constraints

For now, we focus on FPGAs from the **HPC** perspective.

FPGA As Stencil Accelerators ?



Features:

- Large **on-chip** bandwidth
- Fine-grained pipelining
- Customizable datapath / arithmetic

Drawbacks:

- Small **off-chip** bandwidth
- Difficult to program
- Lower clock frequencies

Design Challenges

At least two problems:

- Increase throughput with parallelization.
  Examples:
  - Multiple PEs.
  - Pipelining.
- Decrease bandwidth occupation
  - Use onchip memory to maximize reuse
  - Choose memory mapping carefully to enable burst accesses

Stencils "Done Right" for FPGAs

Observation:

- Many different strategies exist:
  - Multiple-level tiling
  - Deep pipelining
  - Time skewing
  - ...
- No papers put them all together.

Key features:

- Target **one large** deeply pipelined PE...
  - ...instead of **many small** PEs
- Manage throughput/bandwidth with two-level tiling

Multiple-Level Tiling

Composition of 2+ tiling transformations to account for:

- Memory hierarchies and locality
  - Register, caches, RAM, disks, ...
- Multiple level of parallelism
  - Instruction-Level, Thread-Level, ...

In this work:

1. Inner tiling level: parallelism.
2. Outer tiling level: communication.

Overview of Our Approach

Core ideas:

1. Execute inner, **Datapath-Level** tiles on a *single*, *pipelined* "macro-operator".
   - Fire a new tile execution each cycle.
   - Delegate operator pipelining to HLS.
2. Group DL-tiles into **Communication-Level Tiles** to decrease bandwidth requirements.
   - Store intermediary results on chip.

Outline

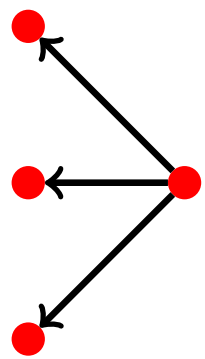
Running Example: Jacobi (3-point, 1D-data)

Simplified code:

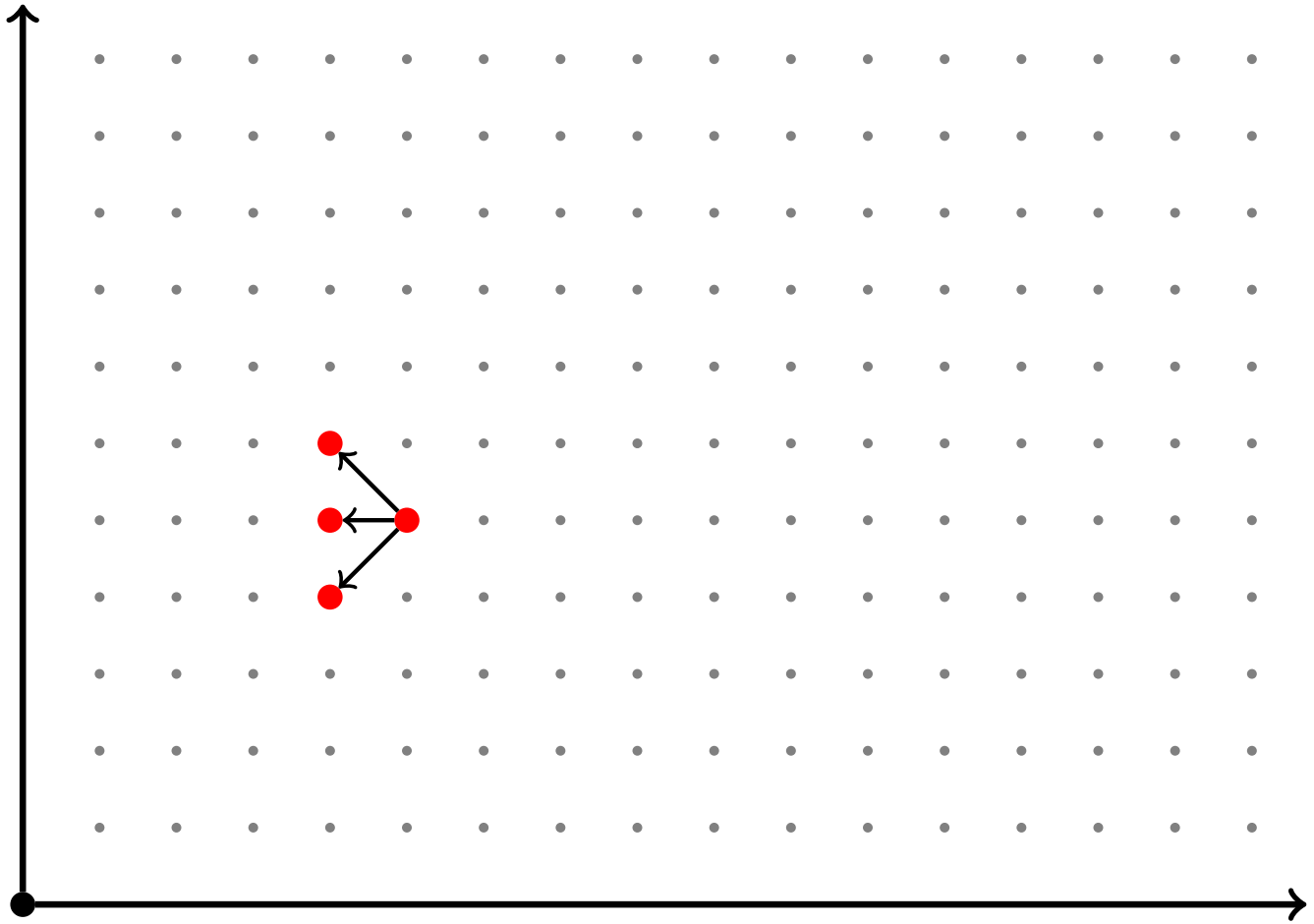
```
for (t=1; t<T; t++)
  for (x=1; x<N-1; x++)
    f[t][x] = (f[t-1][x-1] + f[t-1][x] + f[t-1][x+1])/3;
```

Dependence vectors:

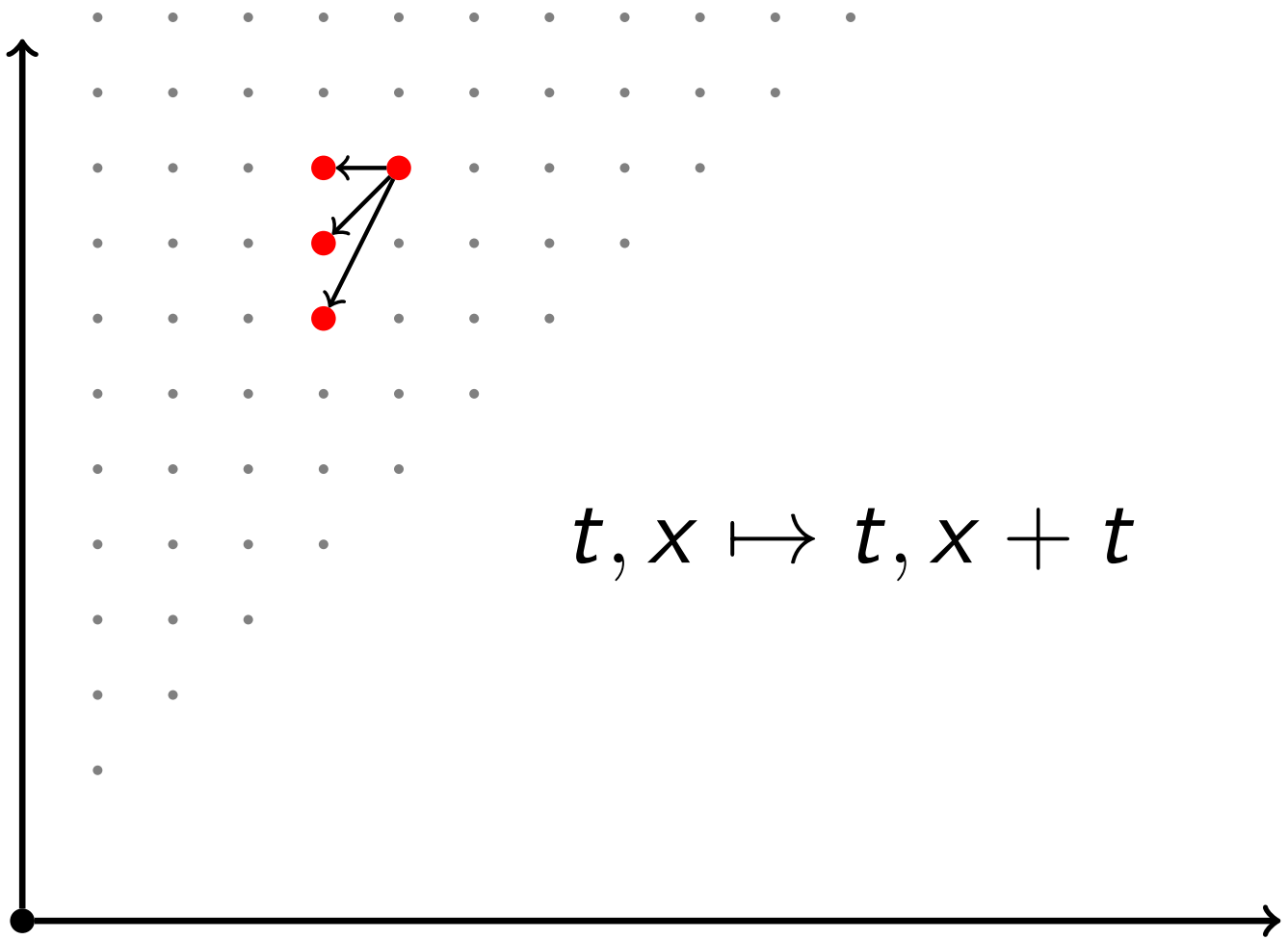
$(-1,-1),(-1,0),(-1,1)$



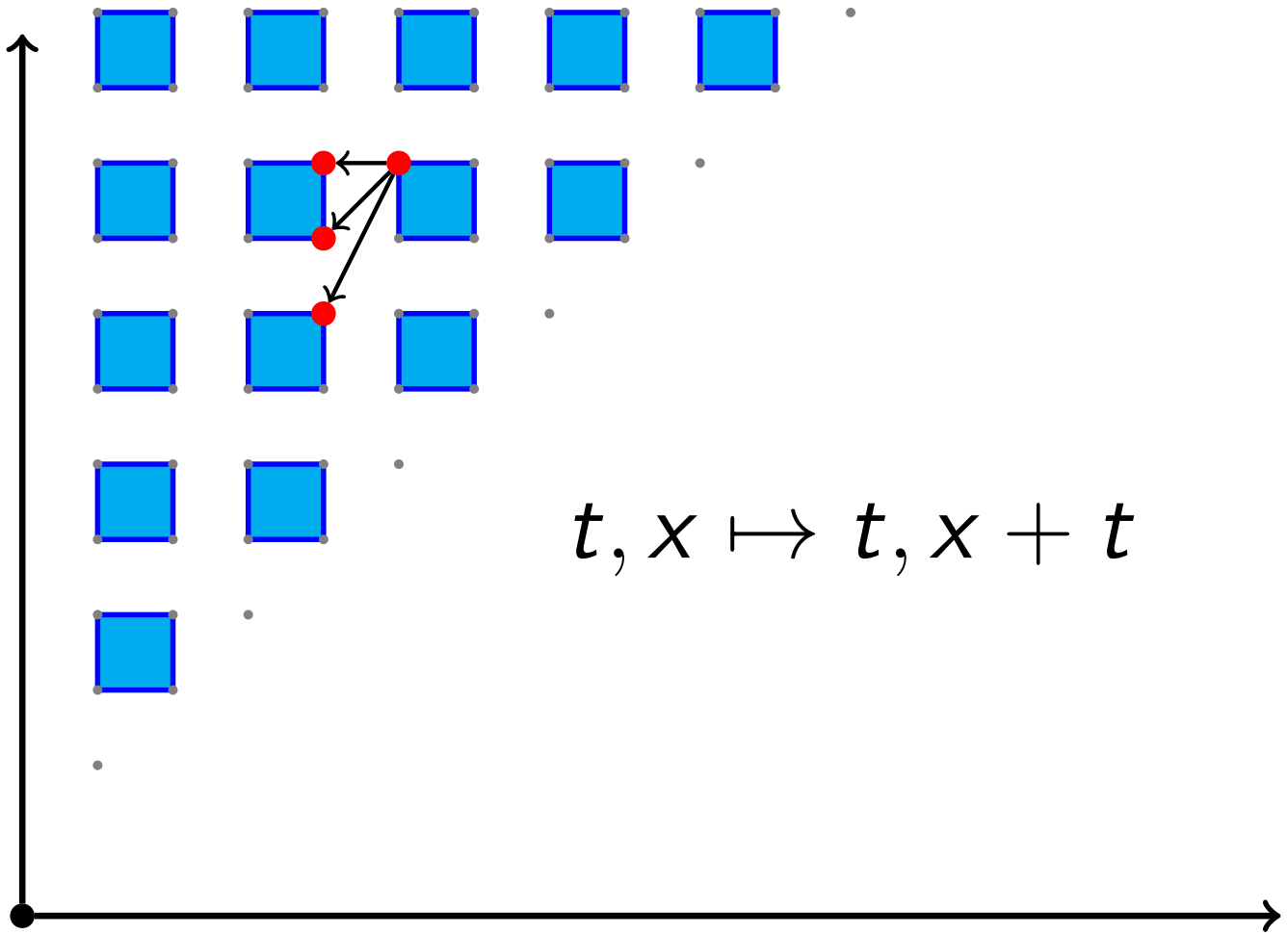
Datapath-Level Tiling



Datapath-Level Tiling



Datapath-Level Tiling



Datapath-Level Tile Operator

```
for (t = ...) {
  #pragma HLS PIPELINE II=1
  for (x = ...) {
    #pragma HLS UNROLL
    for (tt = ...) {
      #pragma HLS UNROLL
      for (xx = ...) {
        int t_ = t+tt, x_ = x+xx-t_;
        f[t_][x_] =
          (f[t_-1][x_-1] + f[t_-1][x_] + f[t_-1][x_+1])/3;
      }
    }
}}
```

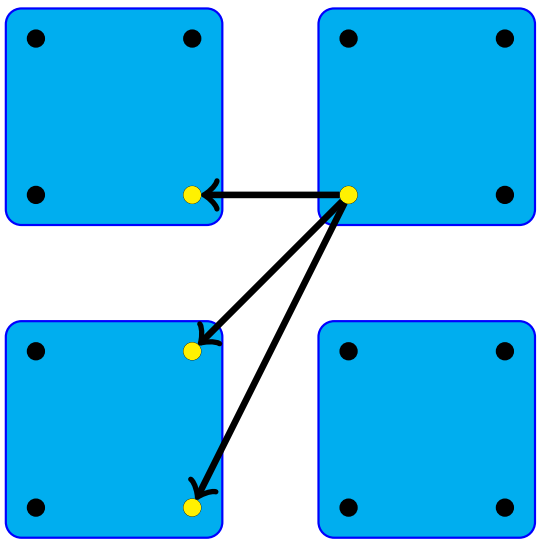
Types of parallelism:

- Operation-Level parallelism (exposed by unrolling).
- Temporal parallelism (through pipelined tile executions).

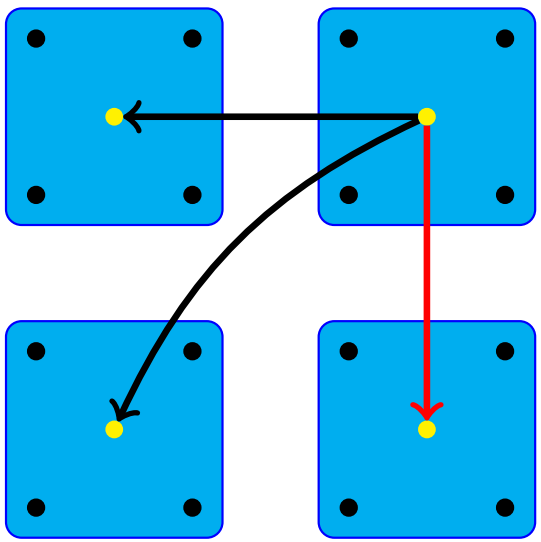
Pipelined Execution

Pipelined execution requires *inter-tile* parallelism.

Original dependences

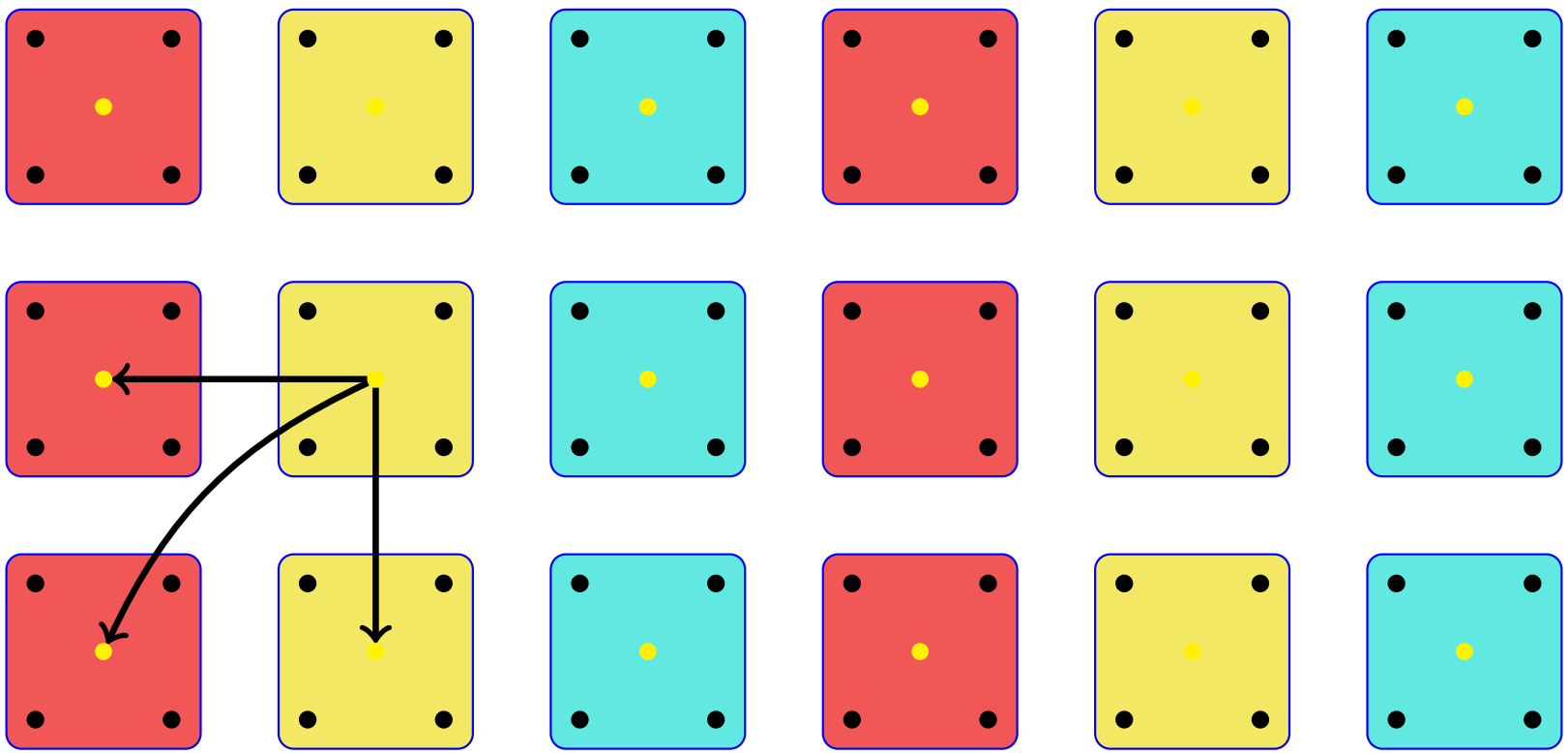


Tile-level dependences



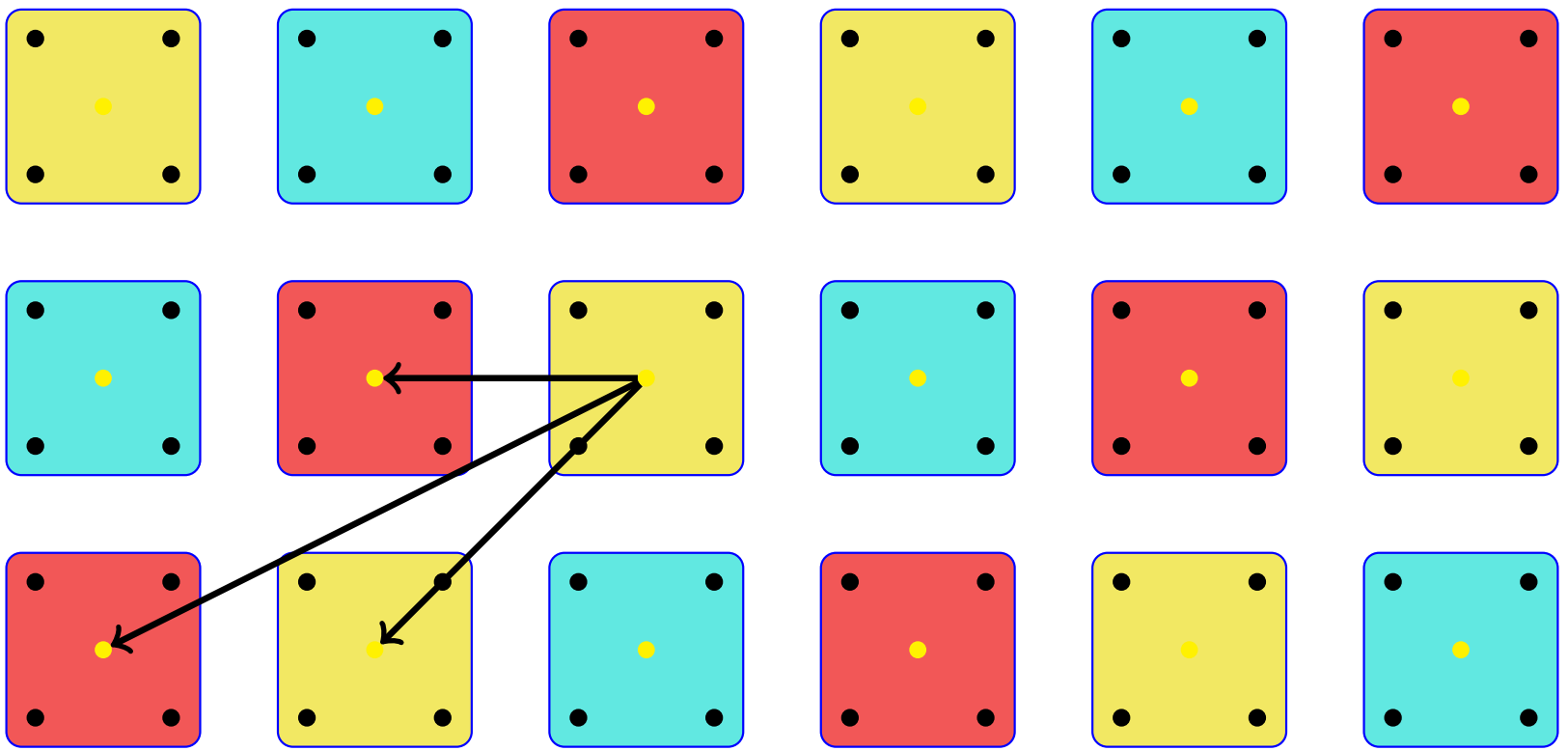
Gauss-Seidel dependences

Wavefronts of Datapath-Level Tiles

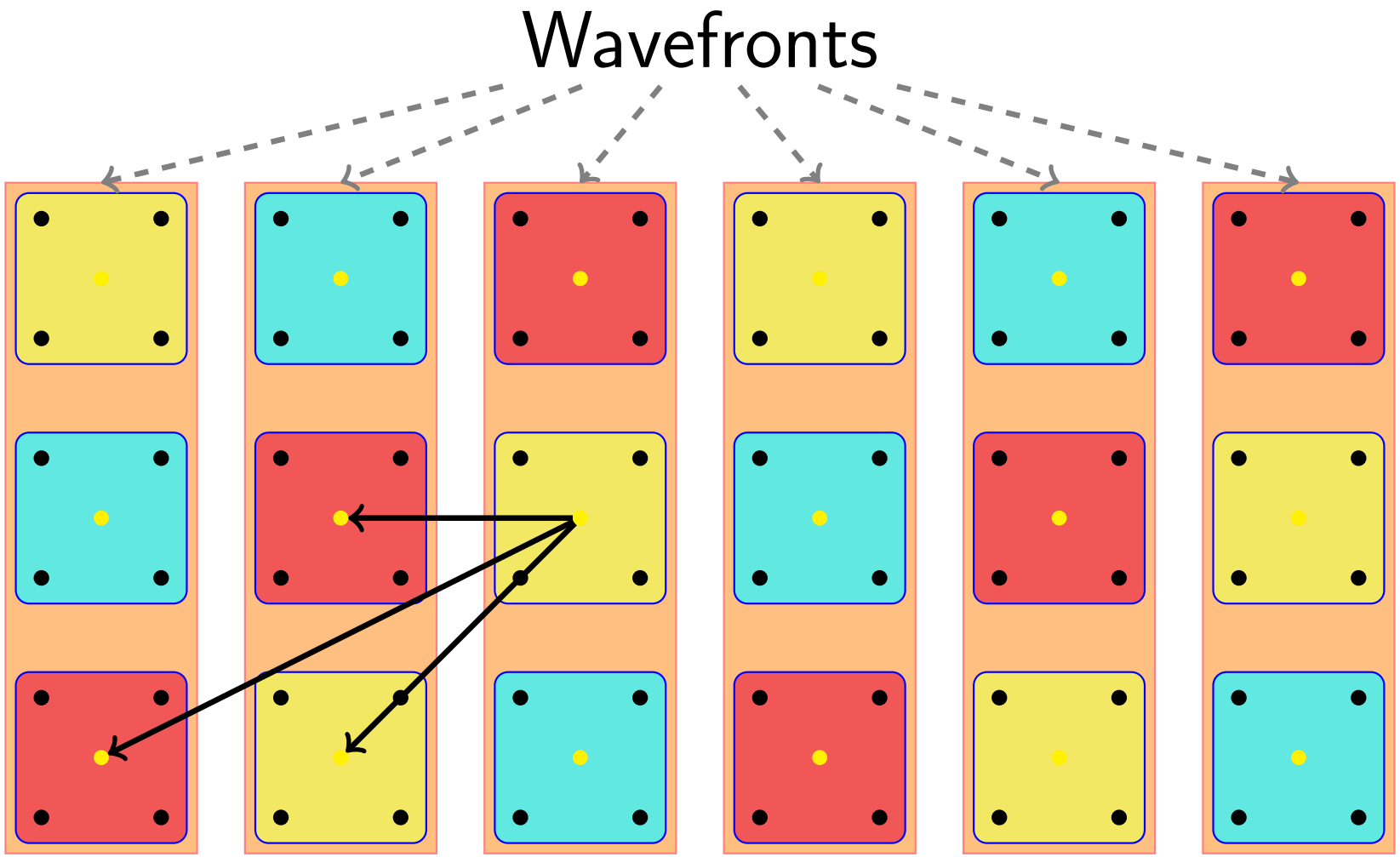


Wavefronts of Datapath-Level Tiles

Skewing: $t, x \mapsto t + x, x$



Wavefronts of Datapath-Level Tiles



Managing Compute/IO Ratio

Problem

Suppose direct pipelining of 2×2 DL-tiles.
At **each** clock cycle:

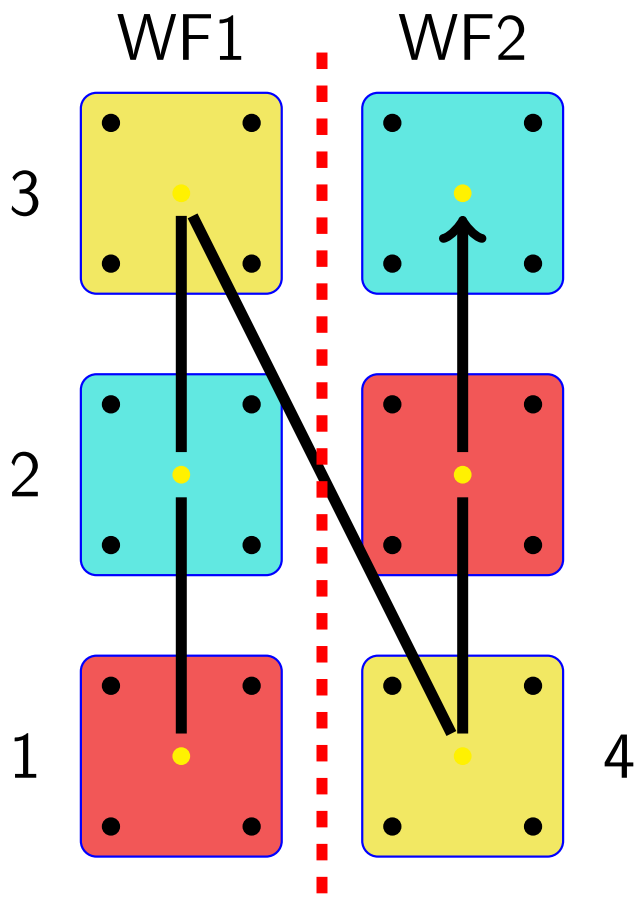
- A new tile enters the pipeline.
- Six 32-bit values are fetched from off-chip memory.

At 100 MHz, bandwidth usage are **19.2 GBps** !

Solution

Use a **second tiling level** to decrease bandwidth requirements.

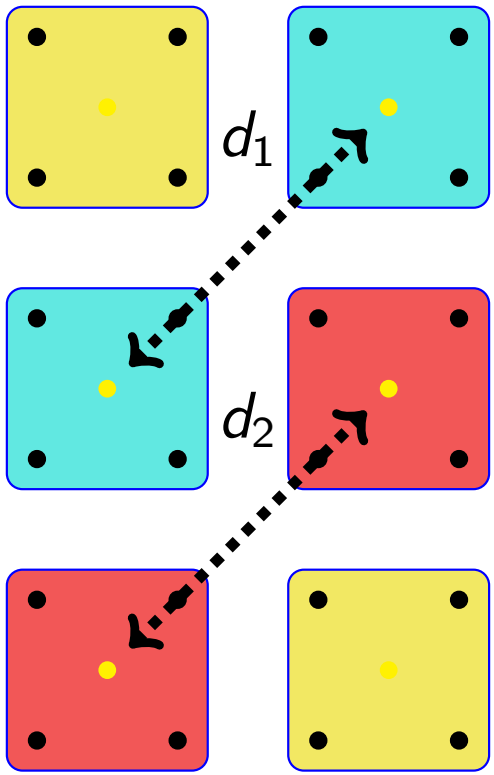
Communication-Level Tiling



Shape constraints:

Size constraints:

Communication-Level Tiling



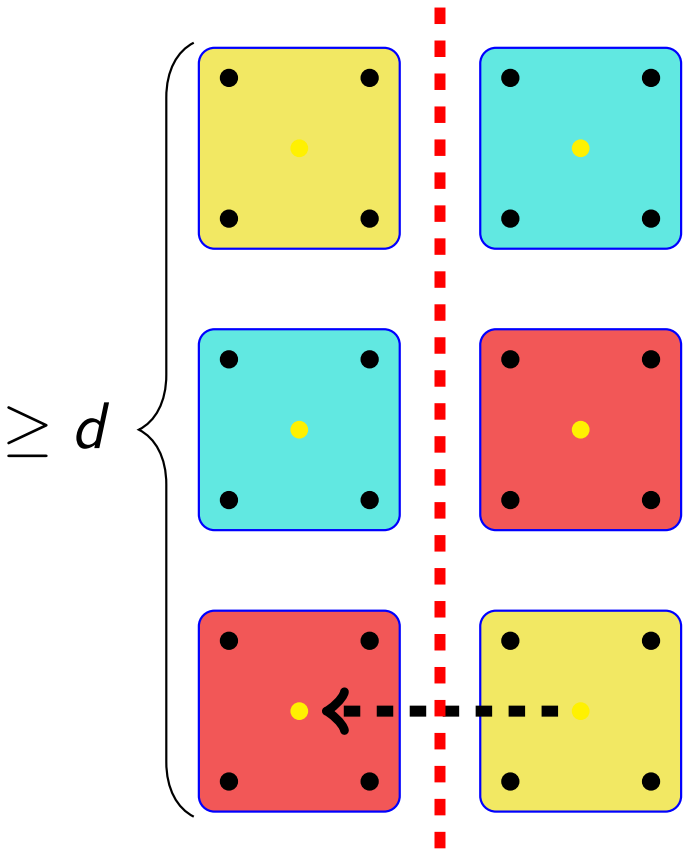
$$d_1 = d_2$$

Shape constraints:

- Constant-height wavefronts
  - Enables use of simple FIFOs for intermediary results

Size constraints:

Communication-Level Tiling

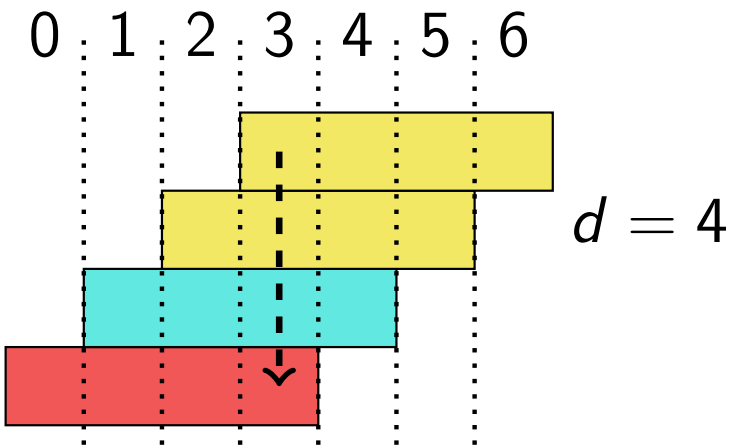


Shape constraints:

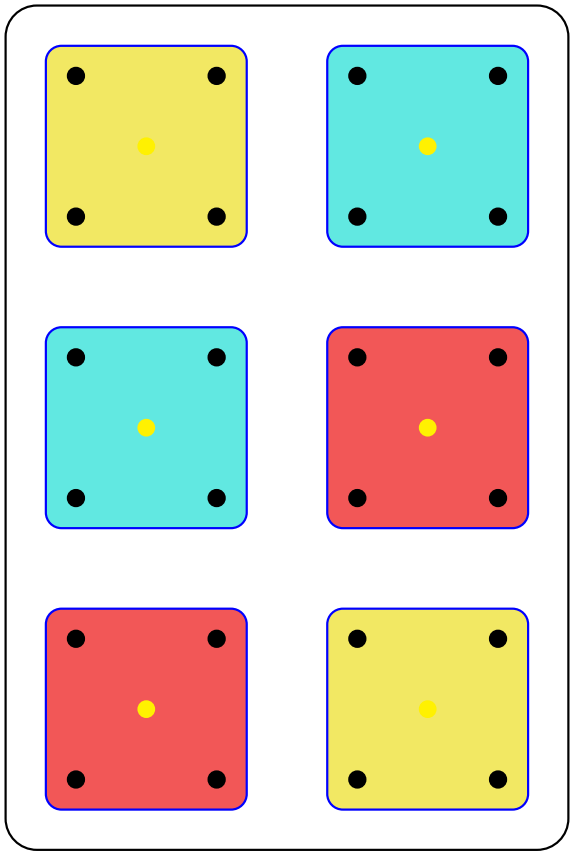
- Constant-height wavefronts
  - Enables use of simple FIFOs for intermediary results

Size constraints:

- Tiles per WF $\geq$ pipeline depth



Communication-Level Tiling



Shape constraints:

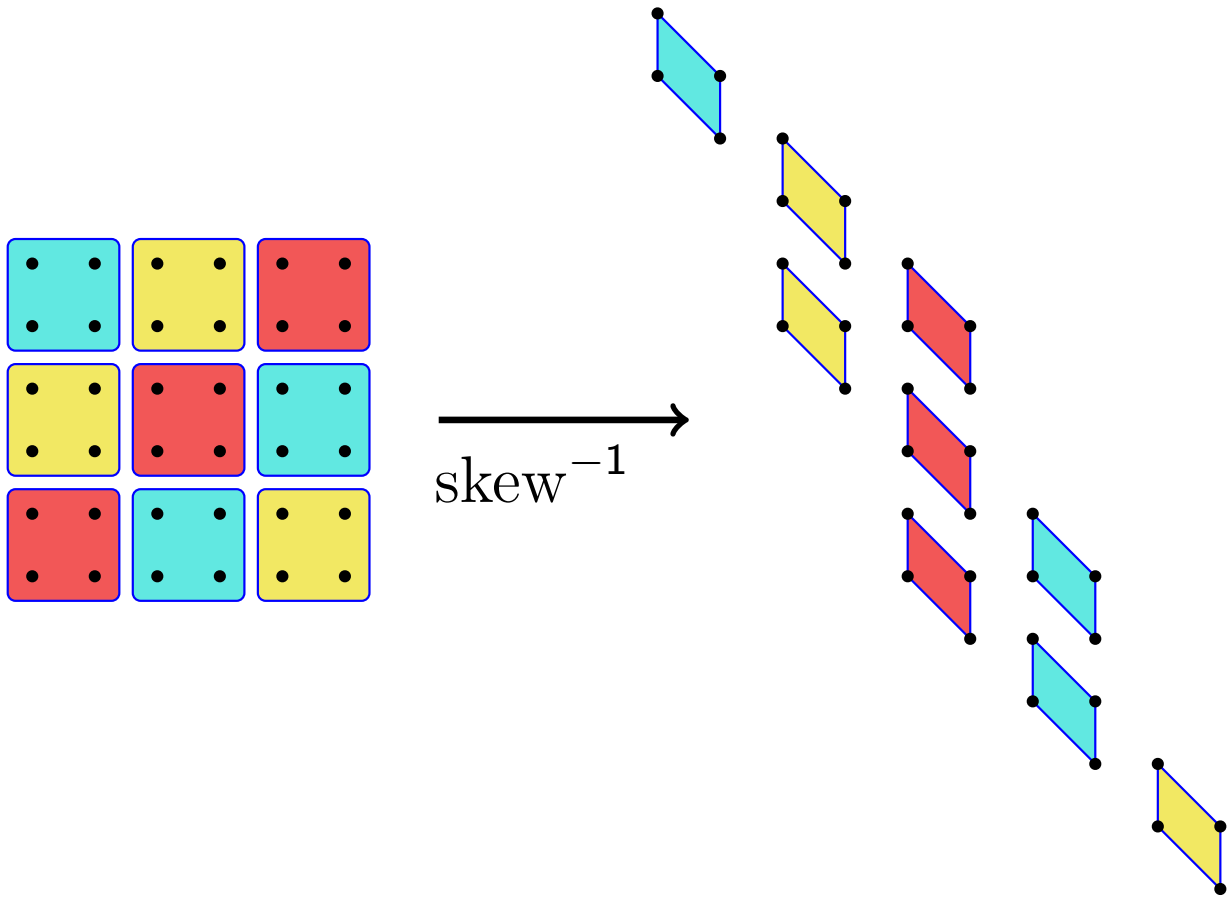
- Constant-height wavefronts
  - Enables use of simple FIFOs for intermediary results

Size constraints:

- Tiles per WF $\geq$ pipeline depth
- BW requirements $\leq$ chip limit
- Size of FIFOs $\leq$ chip limit

Communication-Level Tile Shape

Hyperparallelepipedic (rectangular) tiles satisfy all shape constraints.



Communication

Two aspects:

On-chip Communication

- Between DL-tiles
- Uses FIFOs

Off-chip Communication

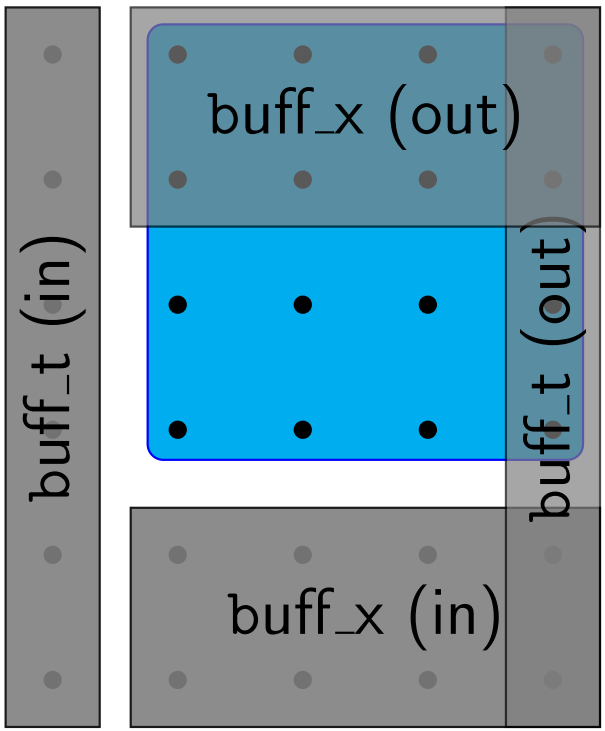
- Between CL-tiles
- Uses memory accesses

On-Chip Communication

We use *Canonic Multi-Projections* (Yuki and Rajopadhye, 2011).

Main ideas:

- Communicate along **canonical axes**.
- Project diagonal dependences on canonical directions.
- Some values are redundantly stored.



Off-Chip Communication

Between CL-Tiles (assuming lexicographic ordering):

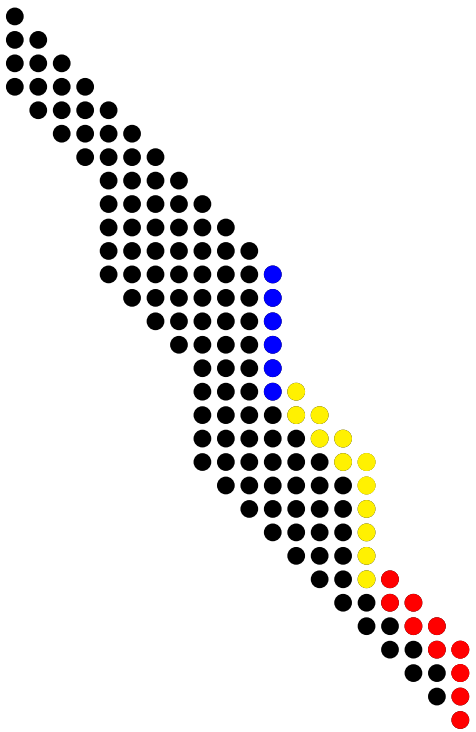
- Data can be reused along the innermost dimension.
- Data from/to other tiles must be fetched/stored **off-chip**.

- Complex shape
- Key for performance: use *burst accesses*
- Maximize contiguity with clever memory mapping

Off-Chip Communication

Between CL-Tiles (assuming lexicographic ordering):

- Data can be reused along the innermost dimension.
- Data from/to other tiles must be fetched/stored **off-chip**.



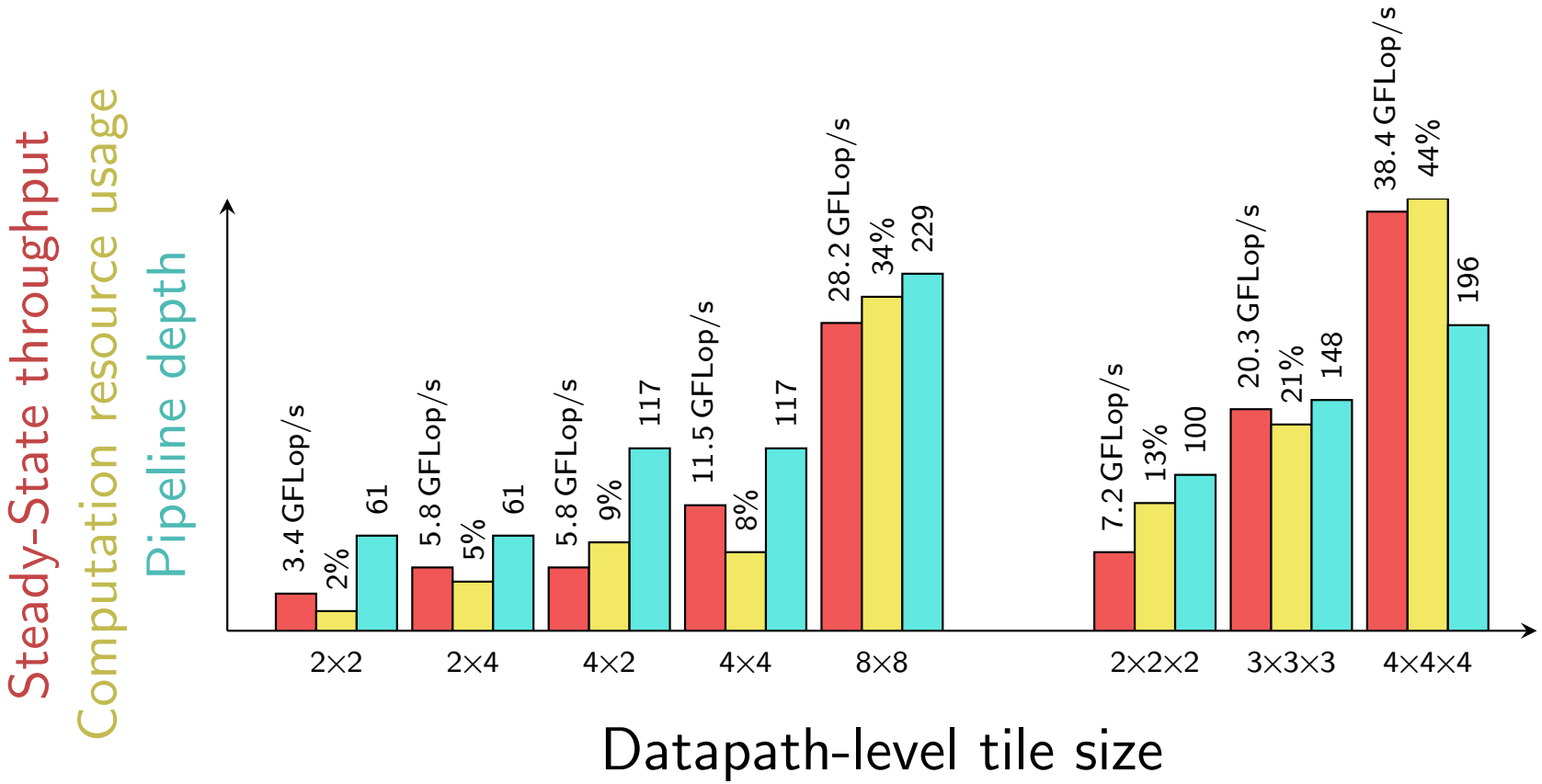
- Complex shape
- Key for performance: use *burst accesses*
- Maximize contiguity with clever memory mapping

Outline

Metrics

- Hardware-related metrics
  - Macro-operator pipeline depth
  - Area (slices, BRAM & DSP)
- Performance-related metrics (at steady state)
  - Throughput
  - Required bandwidth

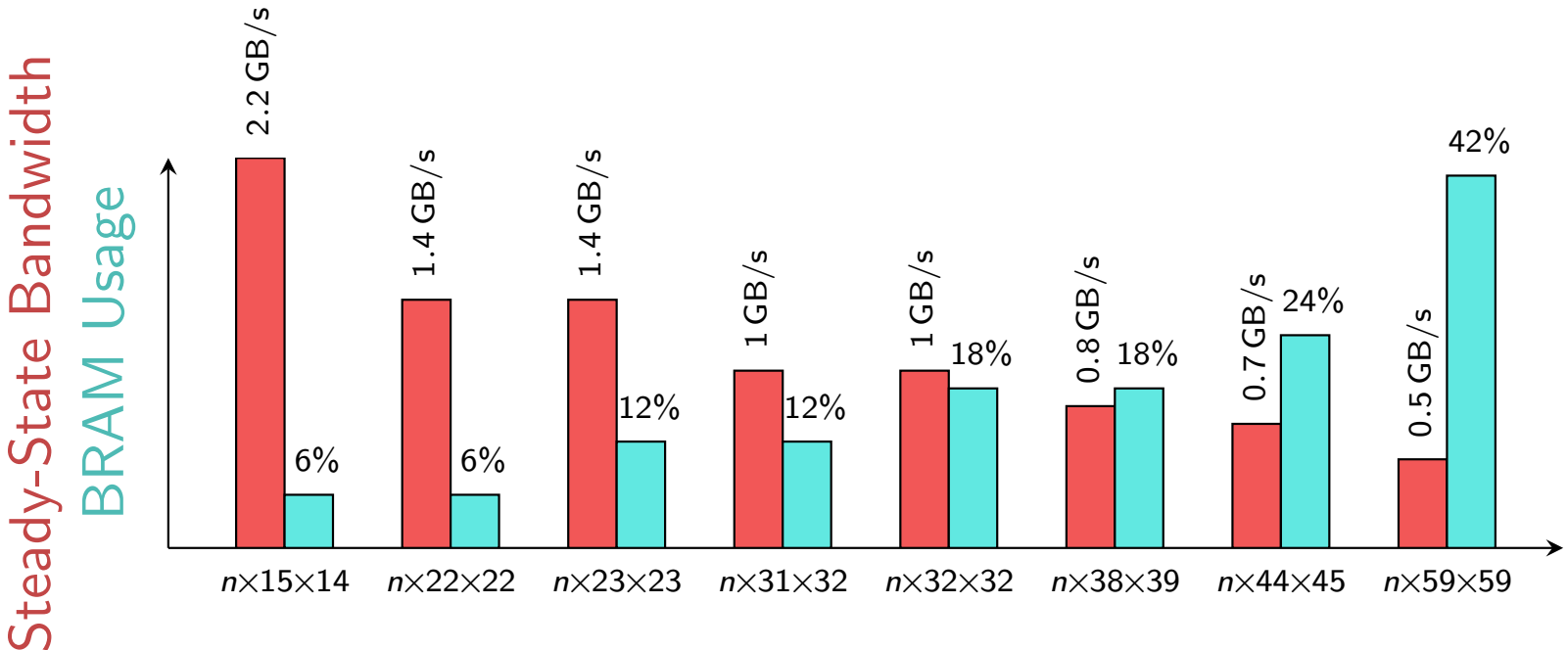
Preliminary Results: Parallelism scalability



Choose DL-tile to control:

- Computational throughput
- Computational resource usage
- Macro-operator latency and pipeline depth

Preliminary Results: Bandwidth Usage Control



Enlarging CL-tiles :

- Does **not** change throughput
- Reduces bandwidth requirements
- Has a low impact on hardware resources

Outline

Related Work

- Hardware implementations:
  - Many ad-hoc / naive architectures
  - Systolic architectures (LSGP)
  - PolyOpt/HLS (Pouchet et al., 2013)
    - Tiling to control compute/IO balance
  - Alias et al., 2012
    - Single, pipelined operator
    - Innermost loop body only
- Tiling method:
  - "Jagged Tiling" (Shrestha et al., 2015)

Outline

Future Work

- Finalize implementation
- Beyond Jacobi
- Exploring other number representations:
  - Fixed-point
  - Block floating-point
  - Custom floating-point
- Hardware/software codesign
- ...

Conclusion

- Design template for FPGA stencil accelerators
- Two levels of control:
  - Throughput
  - Bandwidth requirements
- Maximize use of pipeline parallelism

Thank You

Questions ?