CUDA DGEMM
TILED MATRIX MULTIPLICATION WITH CUDA

25. APRIL 2018 I JOCHEN KREUTZ

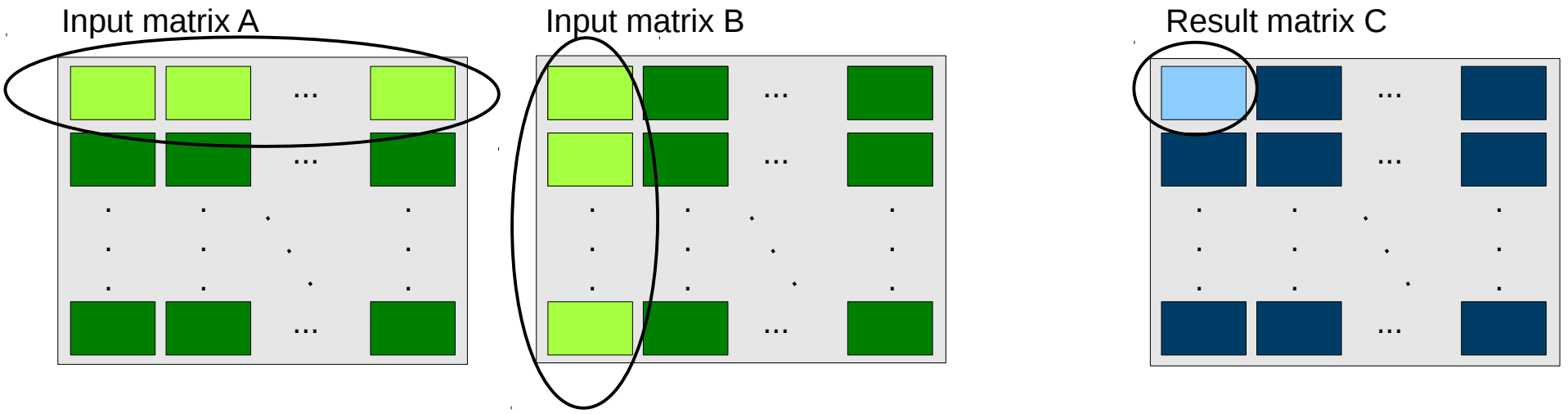
JÜLICH
Forschungszentrum

OVERVIEW

- Tiled matrix multiplication algorithm
- Cuda implementation with and without streams
- Using multi-GPUs and streams

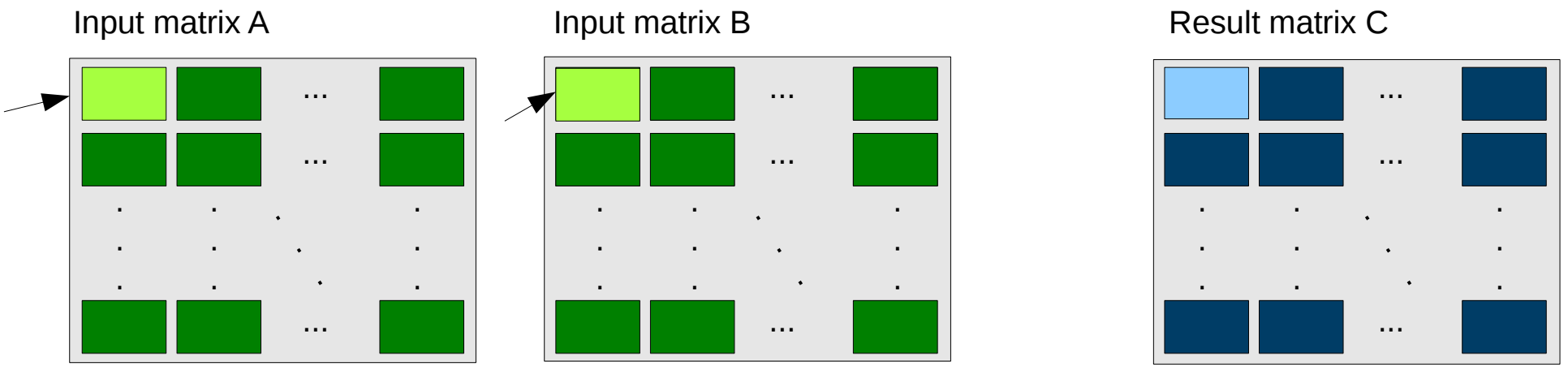
CUDAStreams/Slides/CUDA_DGEMM_tiled.pdf

TILED MATRIX MULTIPLICATION



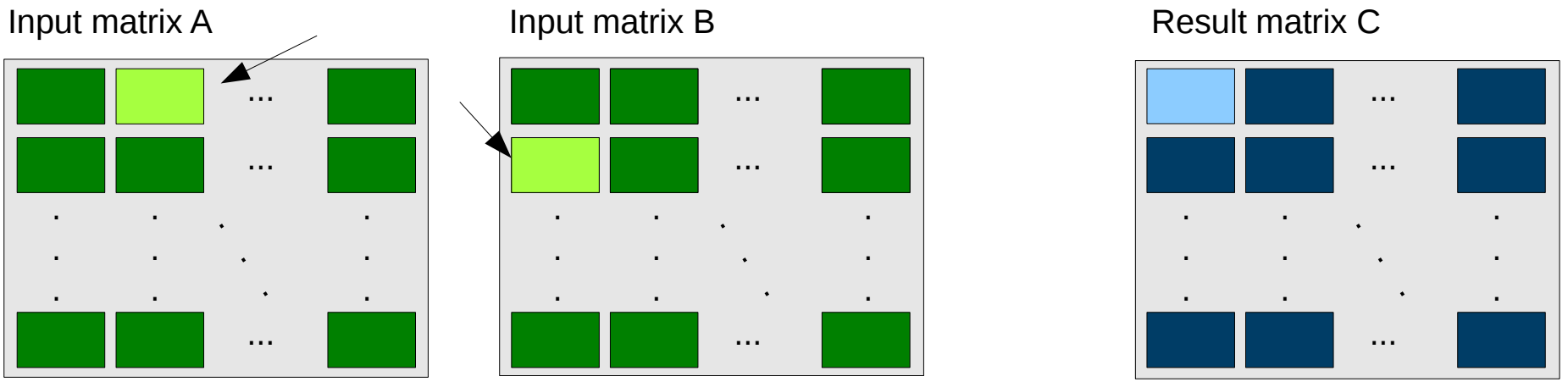
- Split matrices into tiles
- Allows for distributing work onto different streams (and GPUs)

TILED MATRIX MULTIPLICATION



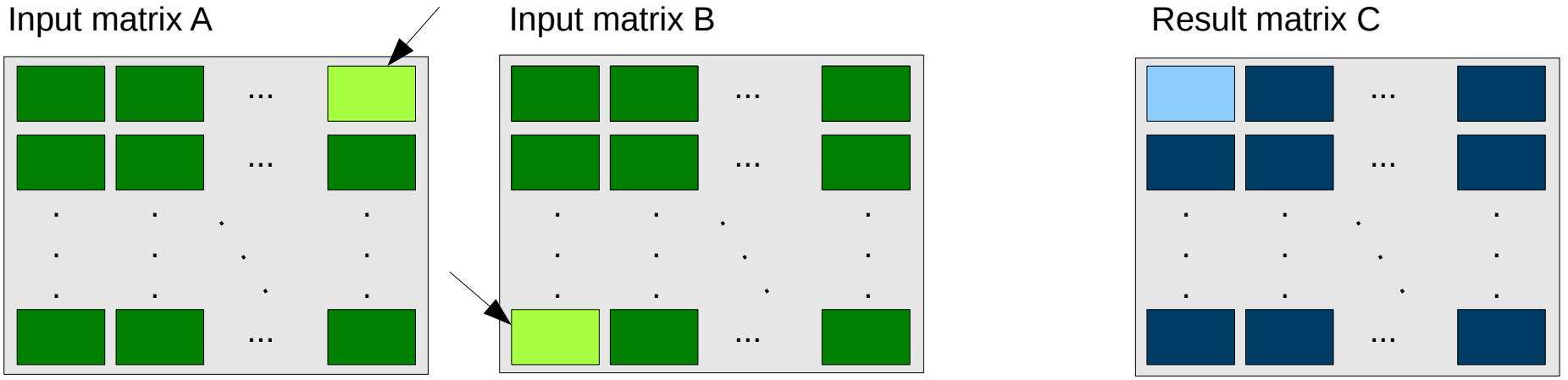
- Do partial (block-wise) computation
- Sum up partial results

TILED MATRIX MULTIPLICATION



- Do partial (block-wise) computation
- Sum up partial results

TILED MATRIX MULTIPLICATION

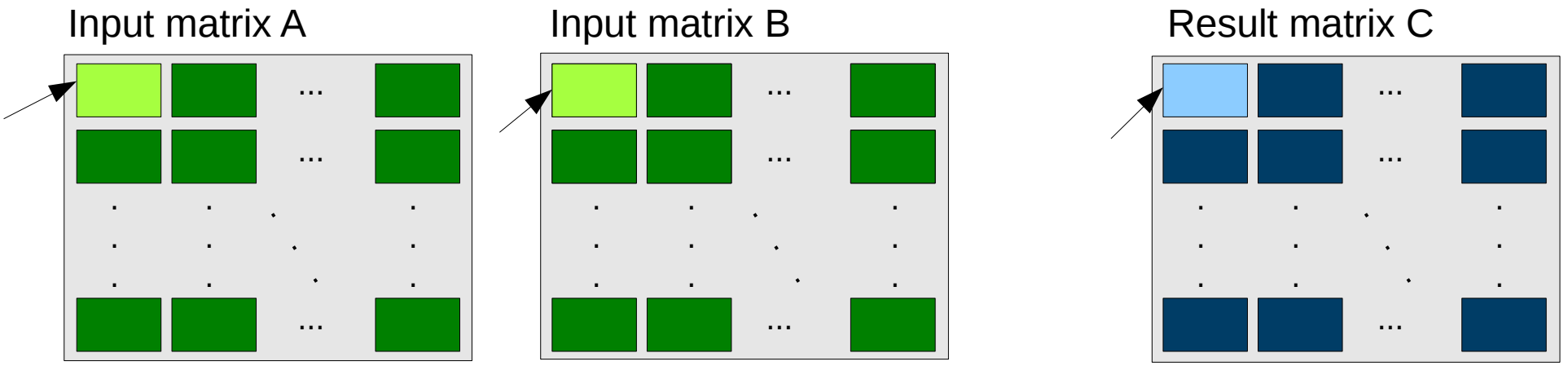


- Do partial (block-wise) computation
- Sum up partial results

TILED MATRIX MULTIPLICATION

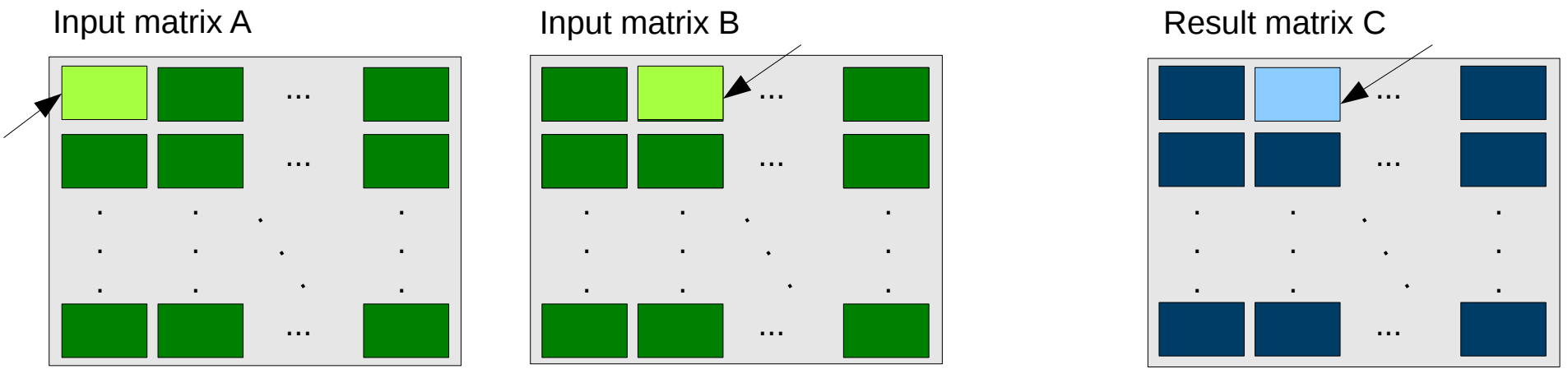
- Change order of computations and run over all tiles of the result matrix in an inner loop
- Do first computations for all tiles in result matrix and then repeat with next tiles of input matrices
- Allows for concurrency in computation of tiles in restul matrix C

TILED MATRIX MULTIPLICATION



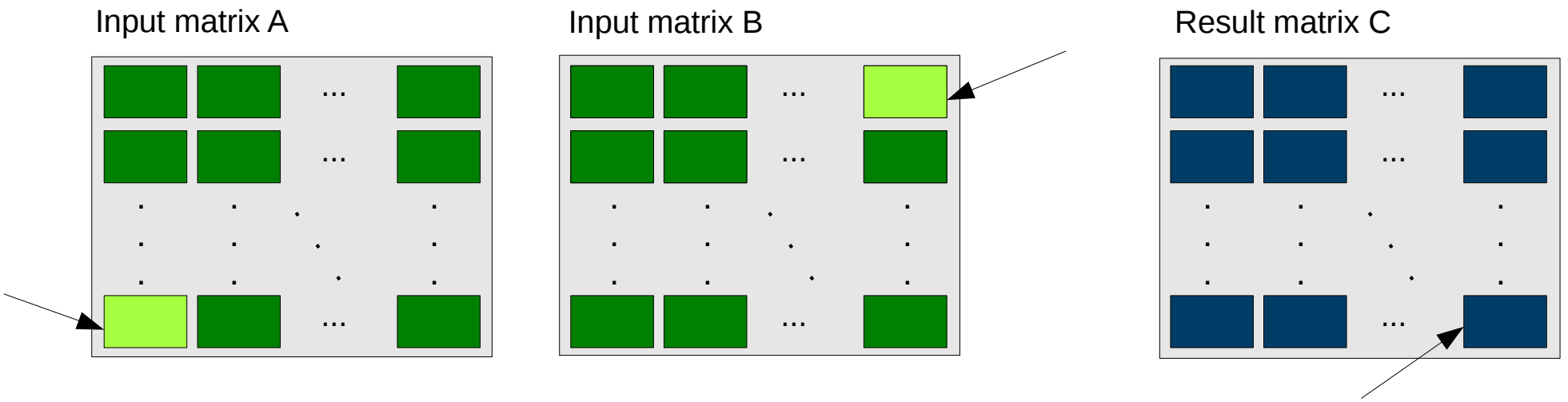
- Change order of computations and run over all tiles of the result matrix with an inner loop
- Do first computations for all tiles in result matrix and then repeat with next tiles of input matrices

TILED MATRIX MULTIPLICATION



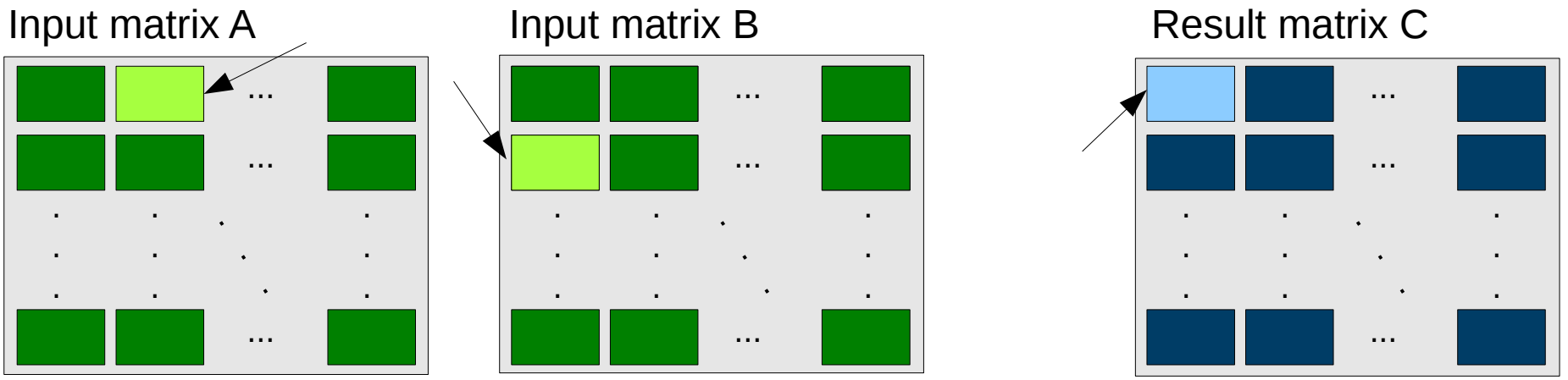
- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

TILED MATRIX MULTIPLICATION



- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

TILED MATRIX MULTIPLICATION



- Change order of computations and run over all tiles of the result matrix in the inner loop
- Do first computations for tiles in result matrix and then proceed to next tiles of input matrices

IMPLEMENTATION

LOOP OVER TILES

```
// loop over inner tile dimension
for ( int iktile = 0; iktile < ntiles; iktile++ ) {

  // loop over row tiles
  for ( int irowtile = 0; irowtile < ntiles; irowtile++ ) {

    // loop over column tiles
    for ( int icoltile = 0; icoltile < ntiles; icoltile++ ) {

    …

    }
  }
}
```

IMPLEMENTATION

- Tiled approach allows to operate large matrices that would not fit into GPU memory as a whole
- For each step only 3 tiles have to be present on the device
- Use pinned memory for tiles to do asynchronous host to device copies and speed up data transfers
- Set beta to 1 in cublasDgemm call to reuse previous calculated results

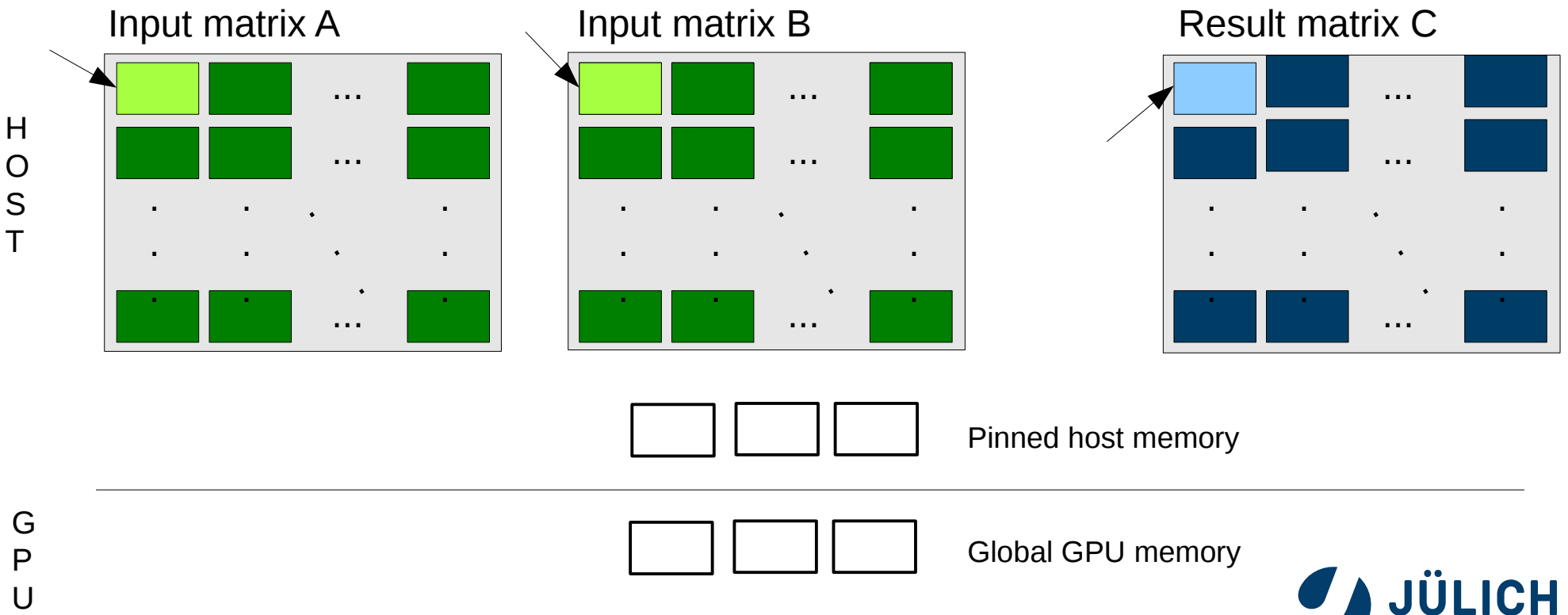
DGEMM COMPUTATION
`C := alpha*op(A)*op(B) + beta*C`

TILED MATRIX MULTIPLICATION - IMPLEMENTATION

- Workflow:
  - Init data (elements of result matrix C have to be set to 0)
  - Loop over tiles in input matrices and over tiles in C

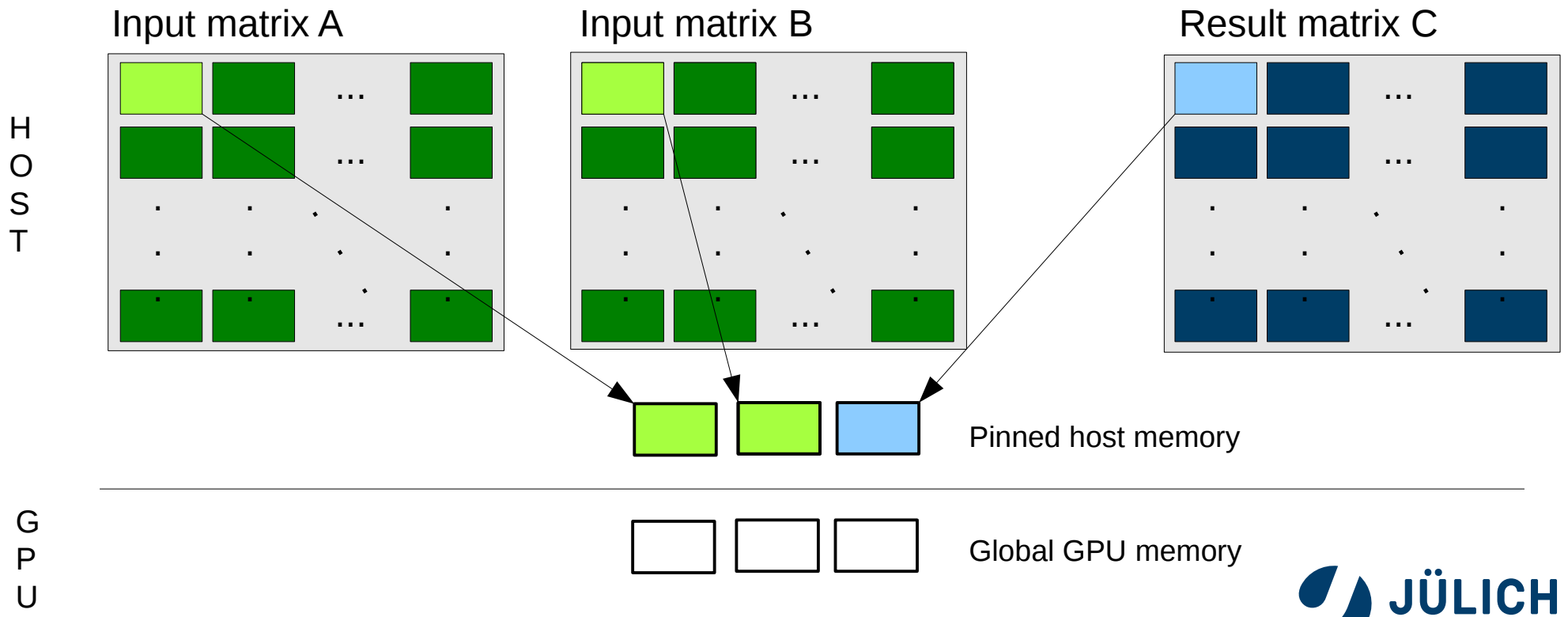
(1) *Read input data (3 tiles) from global matrices to pinned buffers*
(2) Transfer 3 relevant tiles to device
(3) Call `cublasDgemm` *with* `beta = 1`
(4) Read back results from device to pinned buffer
(5) Write back temporary results (1 tile) from pinned host buffer to global result matrix in host memory

TILED MATRIX MULTIPLICATION - IMPLEMENTATION



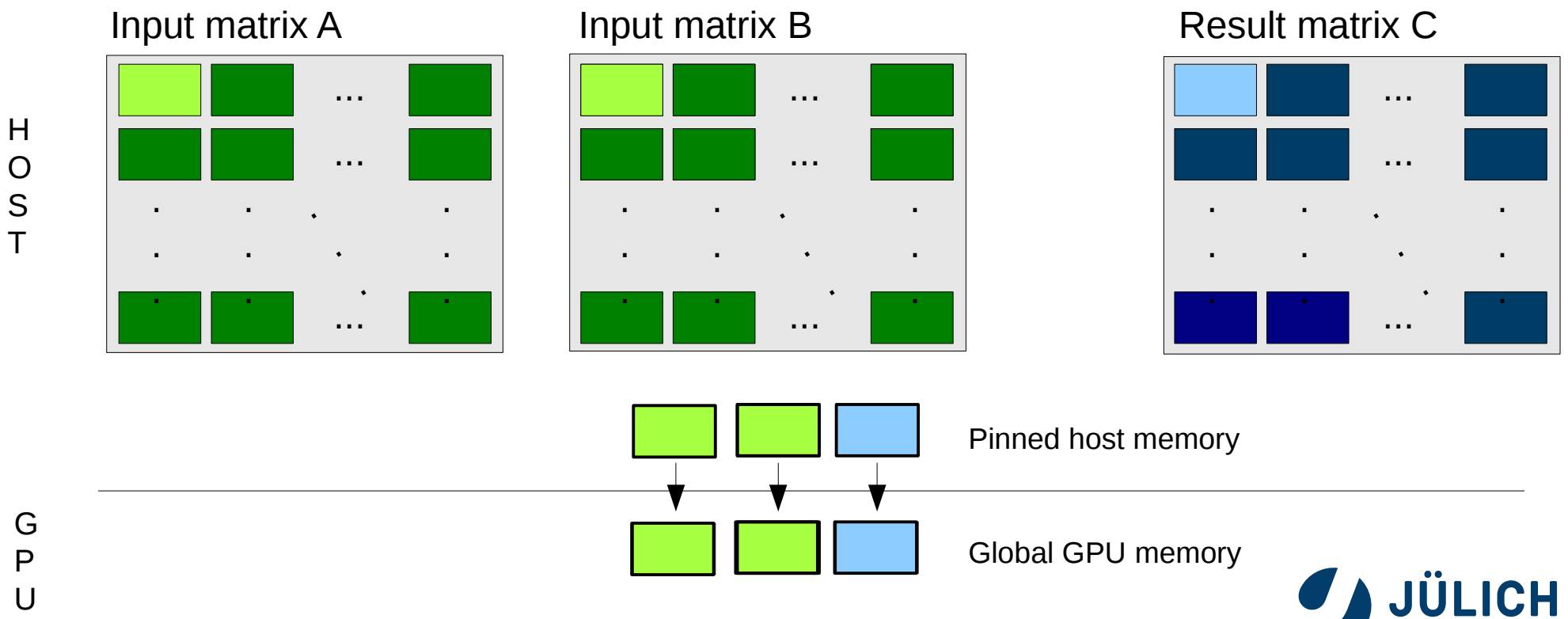
JÜLICH Forschungszentrum

TILED MATRIX MULTIPLICATION - STEP 1

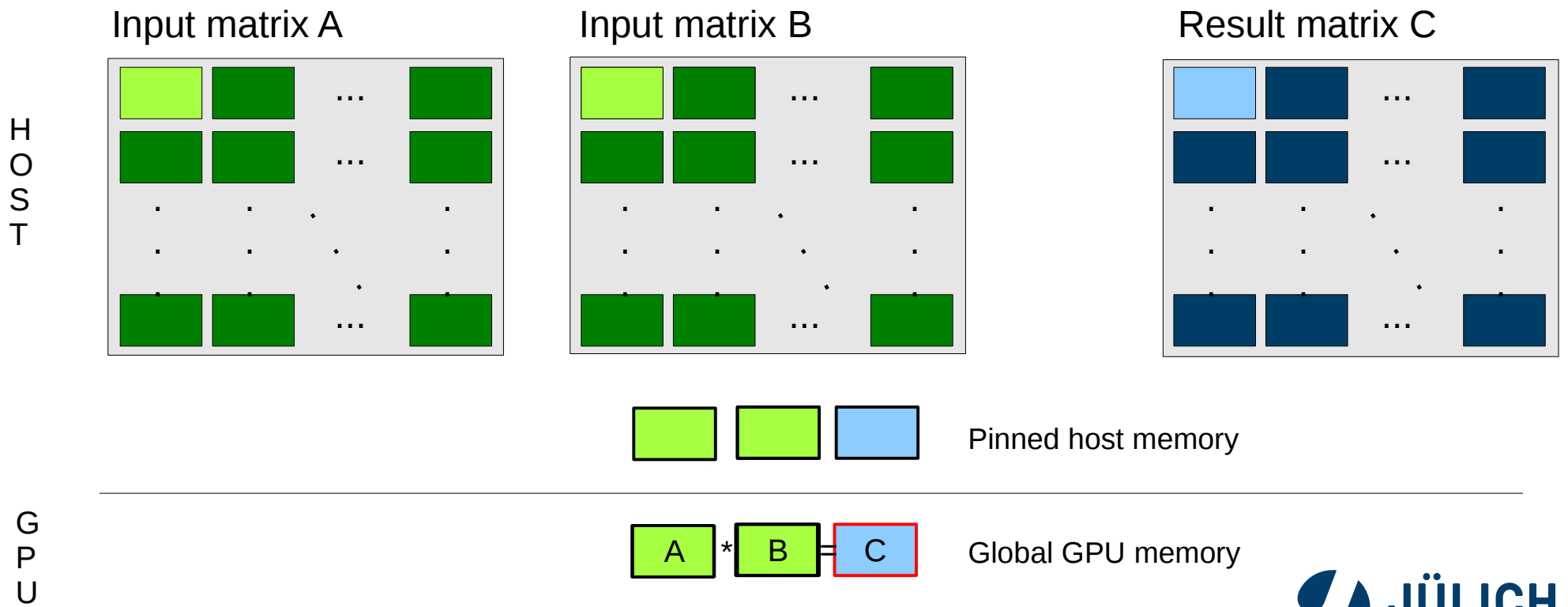


JÜLICH Forschungszentrum

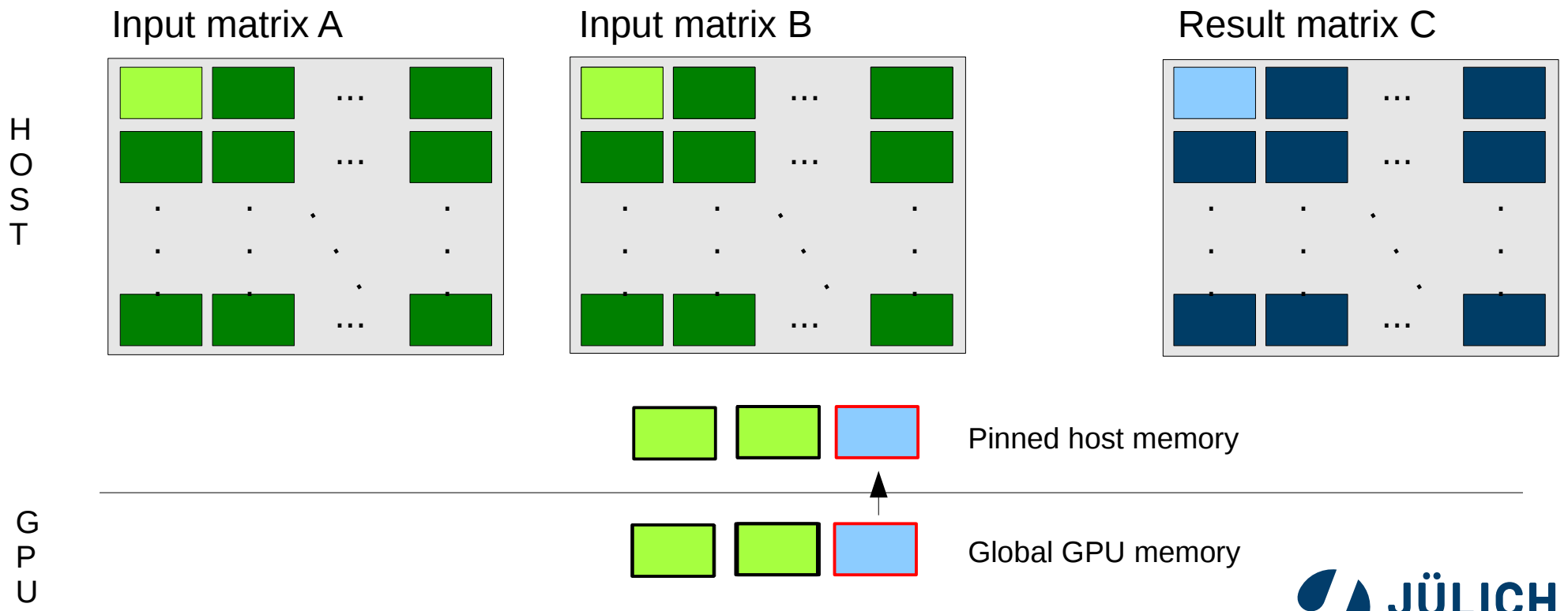
TILED MATRIX MULTIPLICATION - STEP 2



TILED MATRIX MULTIPLICATION – STEP 3

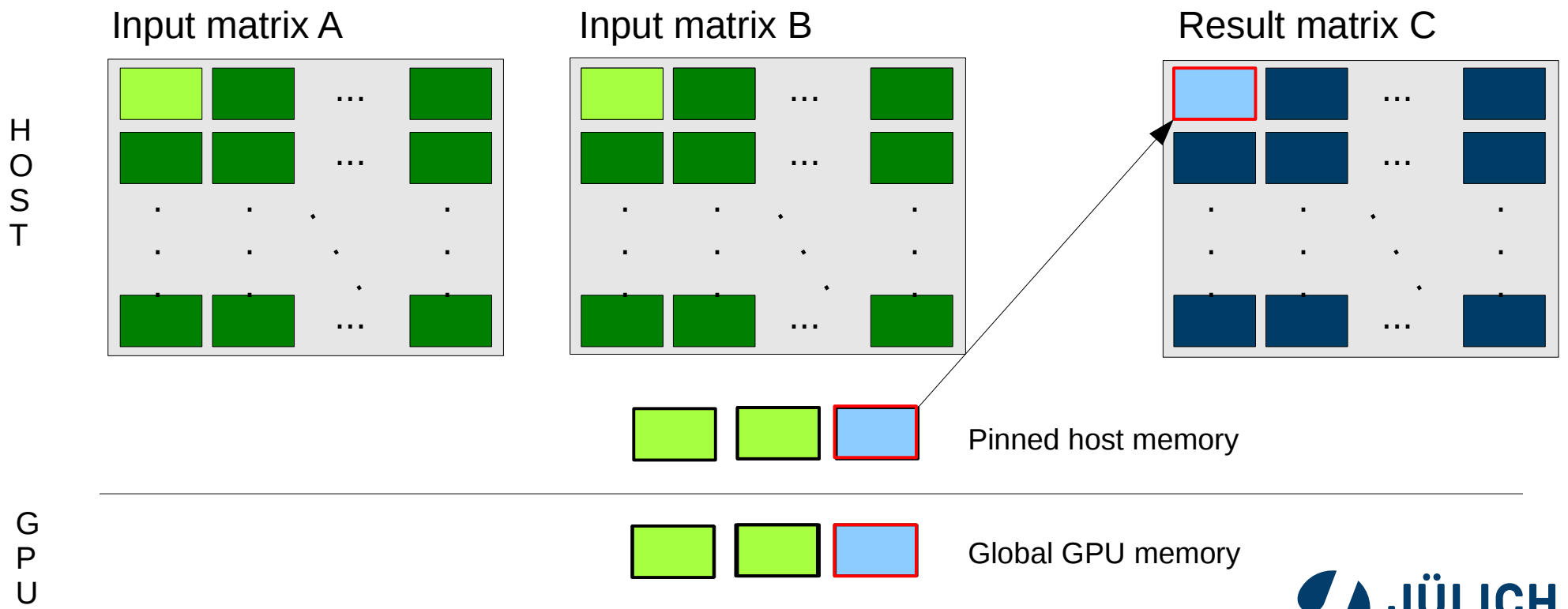


TILED MATRIX MULTIPLICATION – STEP 4



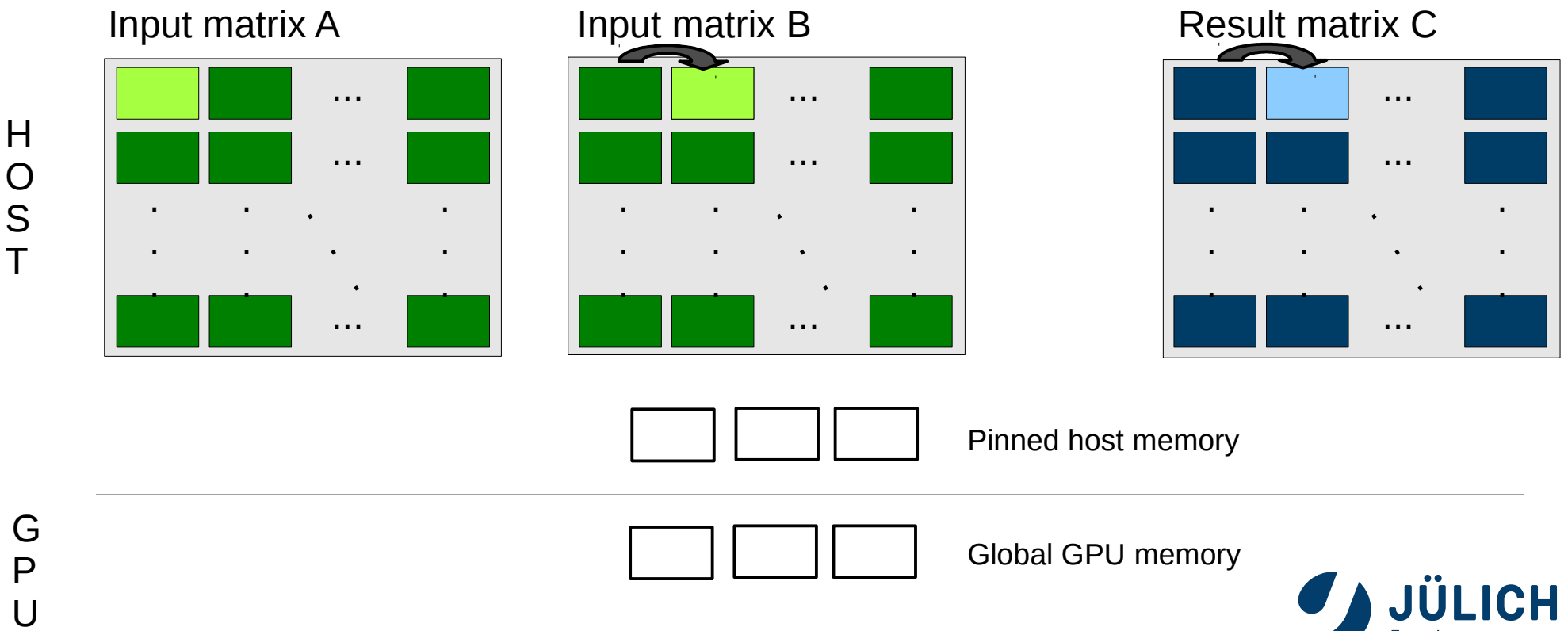
JÜLICH Forschungszentrum

TILED MATRIX MULTIPLICATION – STEP 5



JÜLICH Forschungszentrum

TILED MATRIX MULTIPLICATION
REPEAT STEPS 1 TO 5



EXERCISE

Tiled Matrix Multiplication: task 1

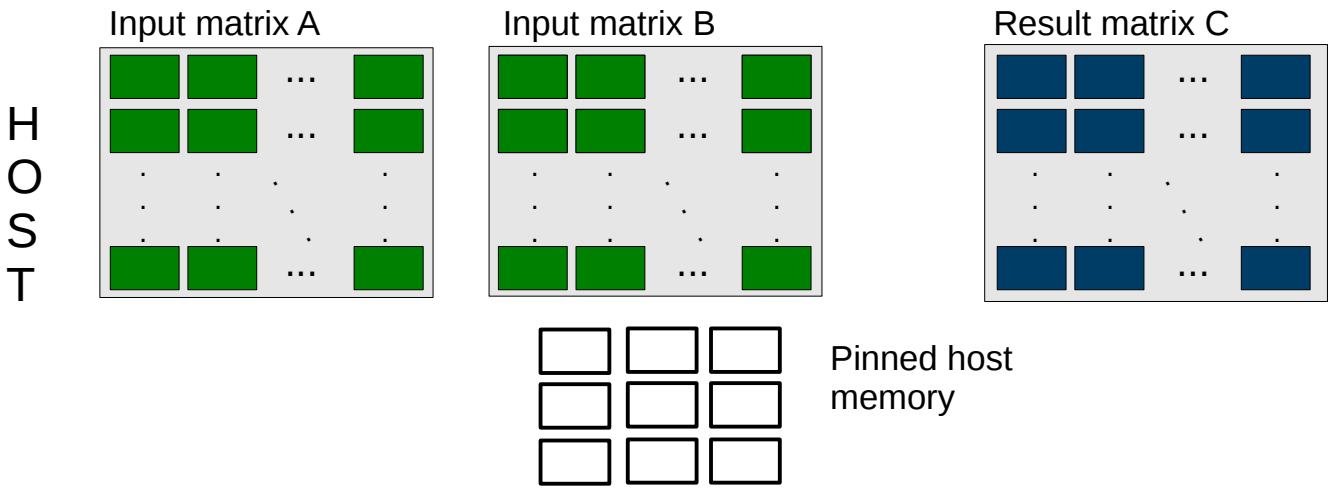
`.../exercises/tasks/Cuda_DGEMM_tiled.cu`

TILED MATRIX MULTIPLICATION – USING STREAMS

- Distribute computation of tiles to different streams
- Use asynchronous data transfers to overlap kernel executions and memory copies
    - Unnecessary data movement can be hidden
    - simplify the implementation

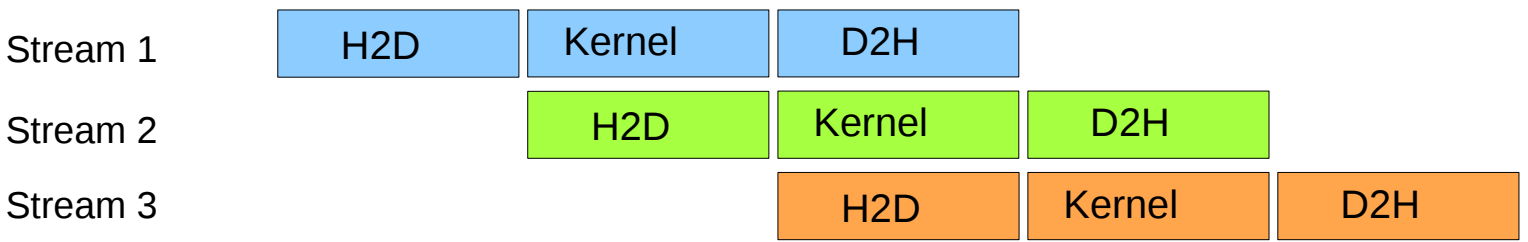
  Each stream will use its own tile buffers (multi buffering)
- Synchronization will be necessary

TILED MATRIX MULTIPLICATION – USING STREAMS: EXAMPLE (3 STREAMS)

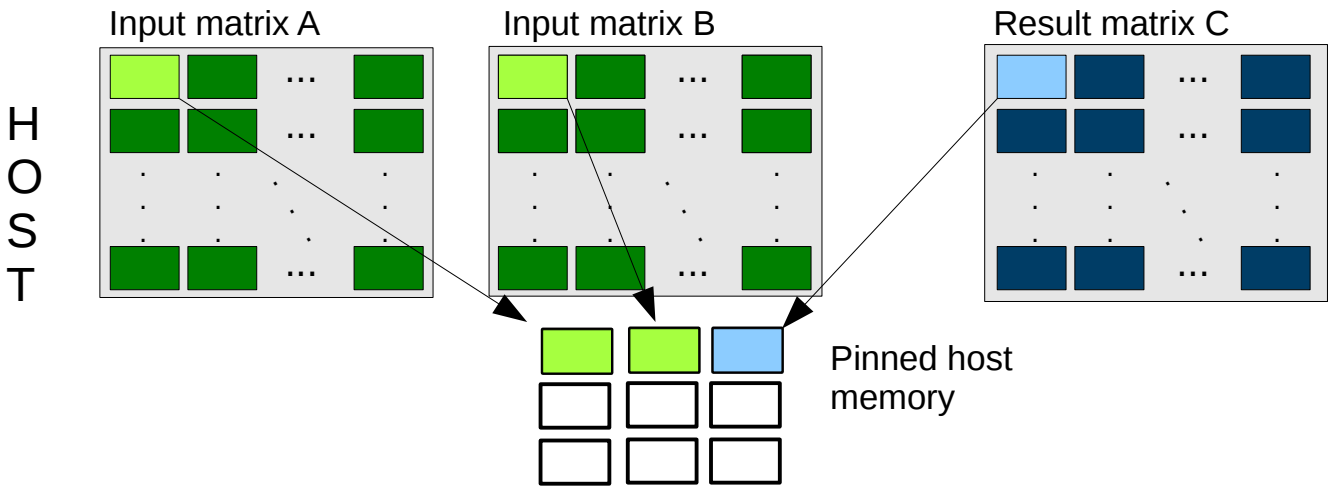


TILED MATRIX MULTIPLICATION – USING STREAMS

- Example: 3 streams
- For every tile:
  - H2D data transfer
  - Kernel execution (dgemm)
  - D2H data transfer

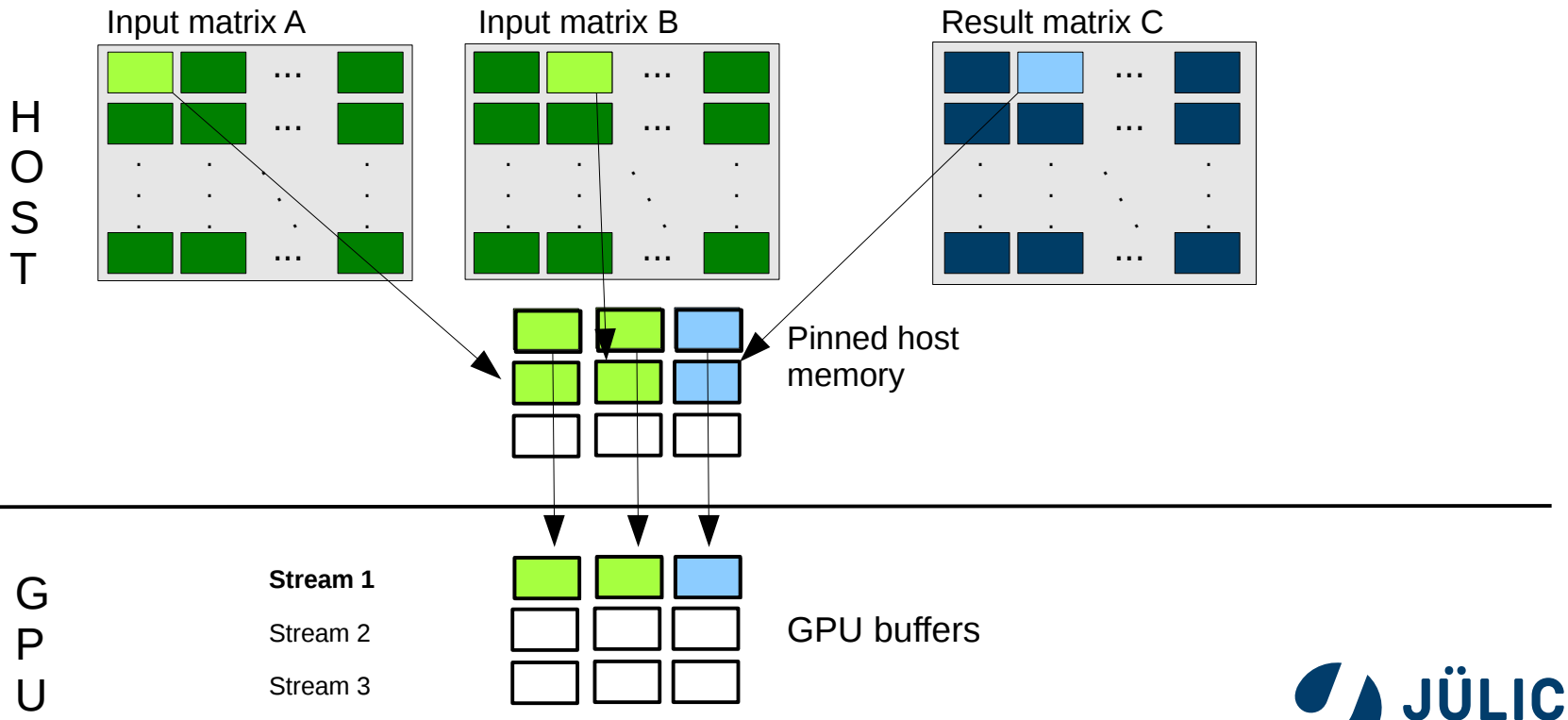


TILED MATRIX MULTIPLICATION – USING STREAMS: EXAMPLE (3 STREAMS)

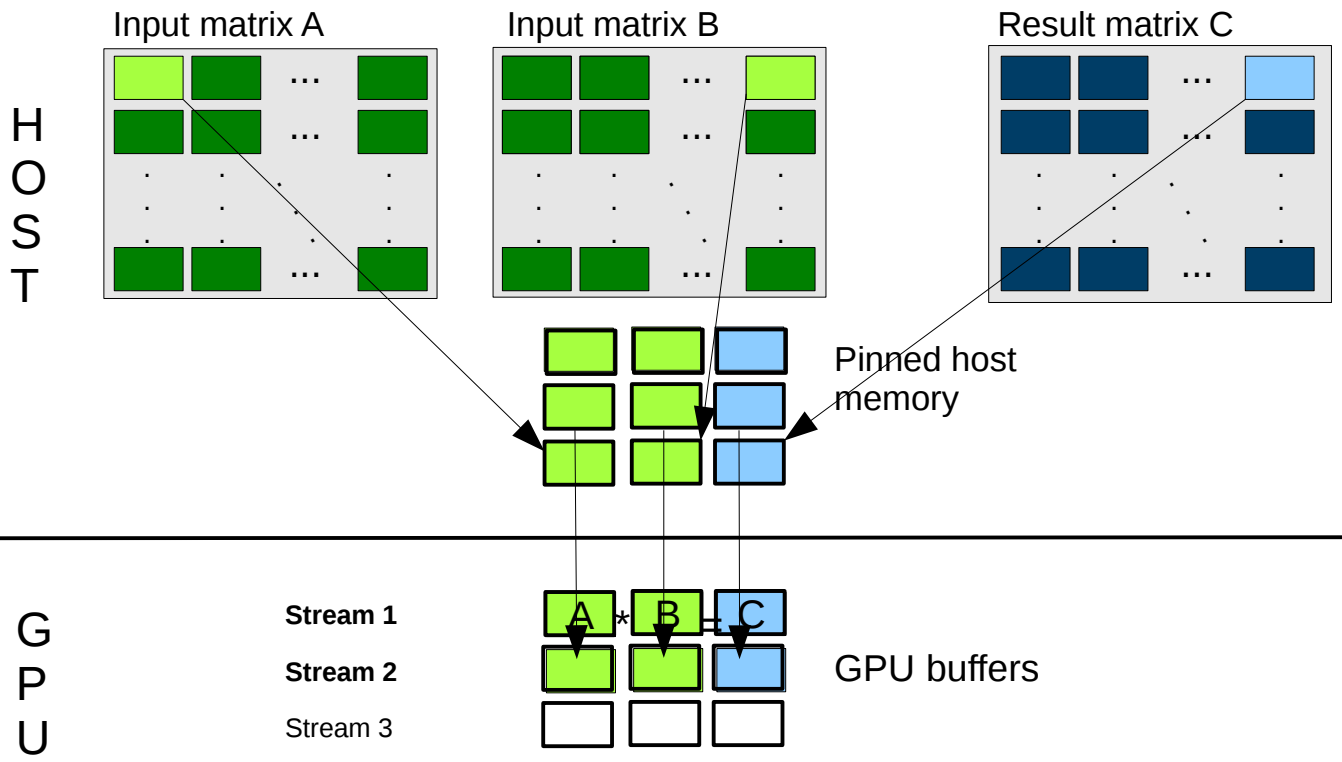


TILED MATRIX MULTIPLICATION – USING STREAMS

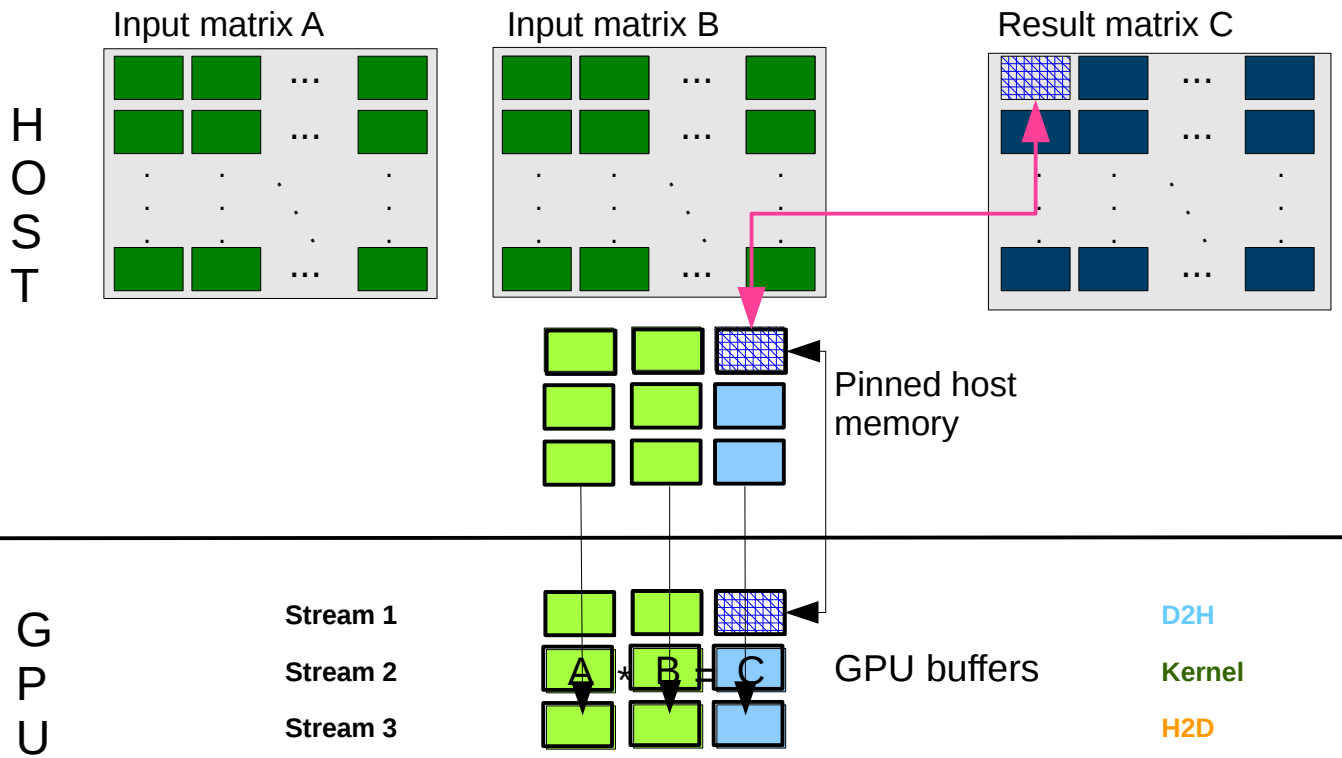
- Example: 3 streams



TILED MATRIX MULTIPLICATION – USING STREAMS: EXAMPLE (3 STREAMS)



TILED MATRIX MULTIPLICATION – USING STREAMS: EXAMPLE (3 STREAMS)



Synchronization needed before writing results back to global matrice in host memory

EXERCISE

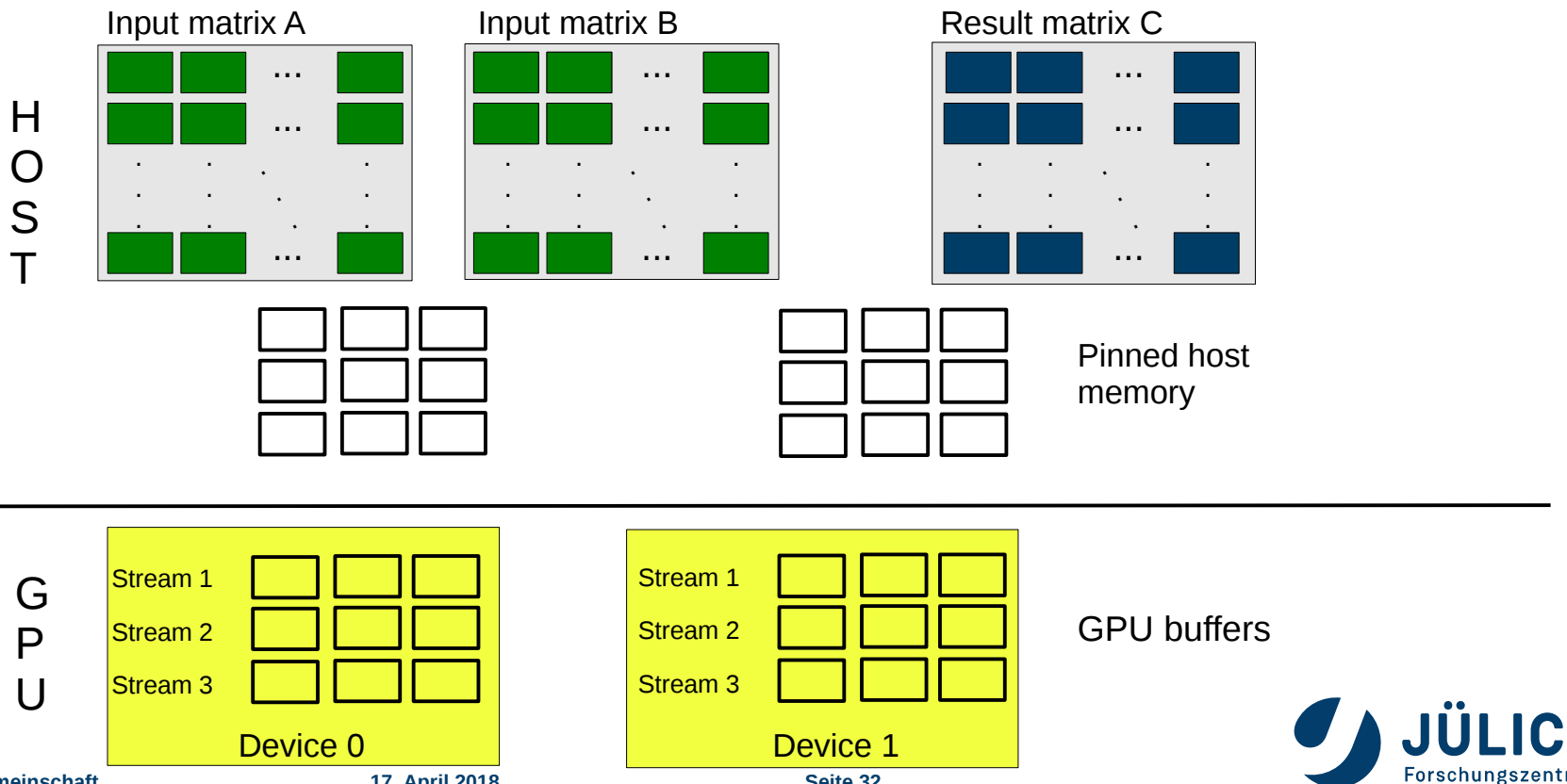
Tiled Matrix Multiplication: task 2

`.../exercises/tasks/Cuda_DGEMM_tiled_streams.cu`

TILED MATRIX MULTIPLICATION – USING MULTIPLE GPUS WITH STREAMS

- Use all GPUs within a node
- Each GPU uses several streams
    - First fill all streams of a GPU then move to next GPU

TILED MATRIX MULTIPLICATION – USING MULTI-GPUS WITH STREAMS: EXAMPLE



EXERCISE

Tiled Matrix Multiplication: task 3

`.../exercises/tasks/Cuda_DGEMM_tiled_streams_multigpu.cu`