Testat 3: GPU Programming

Semester week 10
Submission deadline: *11.05.2023 Rapperswil*
19.05.2023 St Gallen
Submission over Moodle

- Working in a team allowed (max. 2 people). Please put the names on the submission
- What needs to be submitted
  1. Submit your source code
  2. a pdf document reporting the required performance measurements and answers to the questions.

Exercise 10:

Goal:

- Optimize parallel GPU programs with advanced techniques**.**
- Shared memory and synchronization in CUDA resp. OpenCL.

Task 1: Shared Memory und Synchronization

Implement tiled matrix multiplication using shared memory as discussed in the lecture.

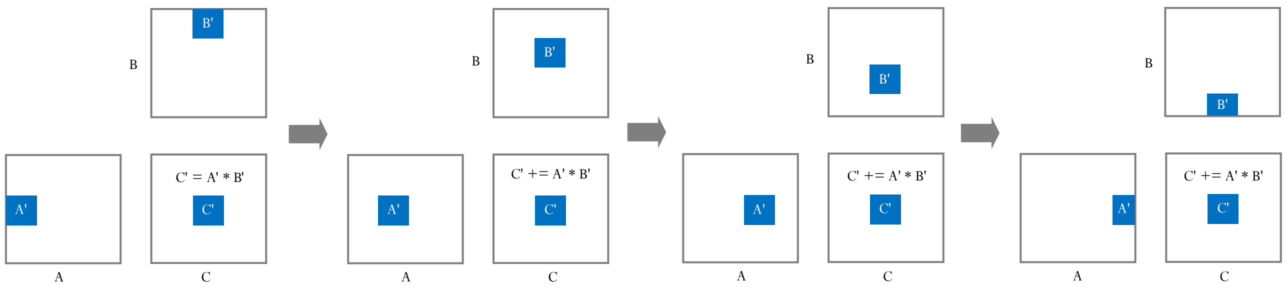
```
__global__
   void matrixMultKernel(float *A, float *B, float *C) {
      __shared__ float Asub[TILE_SIZE][TILE_SIZE];
      __shared__ float Bsub[TILE_SIZE][TILE_SIZE];

      int tx = threadIdx.x, ty = threadIdx.y;
      int col = blockIdx.x * TILE_SIZE + tx;
      int row = blockIdx.y * TILE_SIZE + ty;
      int nofTiles = (A_COLS + TILE_SIZE - 1) / TILE_SIZE;
      //your code

}
```

1. Measure the performance varying the tile size. ***Report your measurements and the best tile size. Is there a relationship between the tile size and the speedup? Explain.***
2. Now Swap the dimensions row and column.
   ```
   int    row    =    blockIdx.x    *    TILE_SIZE    +    tx;
   int    col    =    blockIdx.y    *    TILE_SIZE    +    ty;
   ```
   and compare the run time. Vary the tile size. ***Please report the experiments you performed, your measurements, speedup, plots, and the best size of the tiles for this case.***
3. (Optional) Building on step2, Measure the performance varying the size of the matrix (smaller and bigger), the block size, and the tile size. ***Report your measurements and the best tile size for each matrix. Has the matrix size or block size any influence on the speedup? Explain.***

Recap: With tiled matrix multiplication, each block of matrix C is calculated in several synchronized stages. For each stage, the partial areas of A and B are first loaded into the shared memory in order to reduce repeated accesses in A and B.


B
B'
A'
A
C' = A' * B'
C'
C
B
B'
A'
A
C' += A' * B'
C'
C
B
B'
A'
A
C' += A' * B'
C'
C
B
B'
A'
A
C' += A' * B'
C'
C