CUDA/GPU workshop cheatsheet

Built-in kernel variables

- gridDim.[x,y,z] -> Three dimensional vector containing the dimensions of the grid. This is a constant that is set at kernel launch time. If not set explicitly each dimension defaults to 1.
- blockIdx.[x,y,z] -> Three dimensional vector containing the block index within the grid. This is a dynamic value that depends on which block calls it.
- blockDim.[x,y,z] -> Three dimensional vector containing the dimensions of the thread block. This is set at kernel launch time. If not set explicitly each dimension defaults to 1.
- threadIdx.[x,y,z] -> Three dimensional vector specifying the thread index within the thread block. Dynamic value depending on which thread calls it.

Important Functions

- Kernel Launch
  - `void Kernel_name<<< gridsize, blocksize >>>(arg1,arg2,…);`
- Memory Management
  - `cudaError_t cudaMalloc( void **devPtr, size_t size );`
    - Example: **`cudaMalloc( (void **) &d_c, numbytes );`**
  - `cudaError_t cudaFree( void *devPtr );`
    - Example: **`cudaFree( d_c );`**
  - `cudaError_t cudaMemcpy( void *dst, const void *src, size_t size, enum cudaMemcpyKind kind );`
    - `enum cudaMemcpyKind`
      - `cudaMemcpyHostToDevice`
      - `cudaMemcpyDeviceToHost`
      - `cudaMemcpyDeviceToDevice`
    - Example: **`cudaMemcpy( d_c, c, numbytes,cudaMemcpyHostToDevice);`**
- Error Checking
  - `cudaError_t cudaGetLastError(void);`
  - `char* cudaGetErrorString( cudaError_t code );`
  - **`printf("%s\n", cudaGetErrorString( cudaGetLastError() ) );`**

Hierarchy of Grid->Blocks->Threads

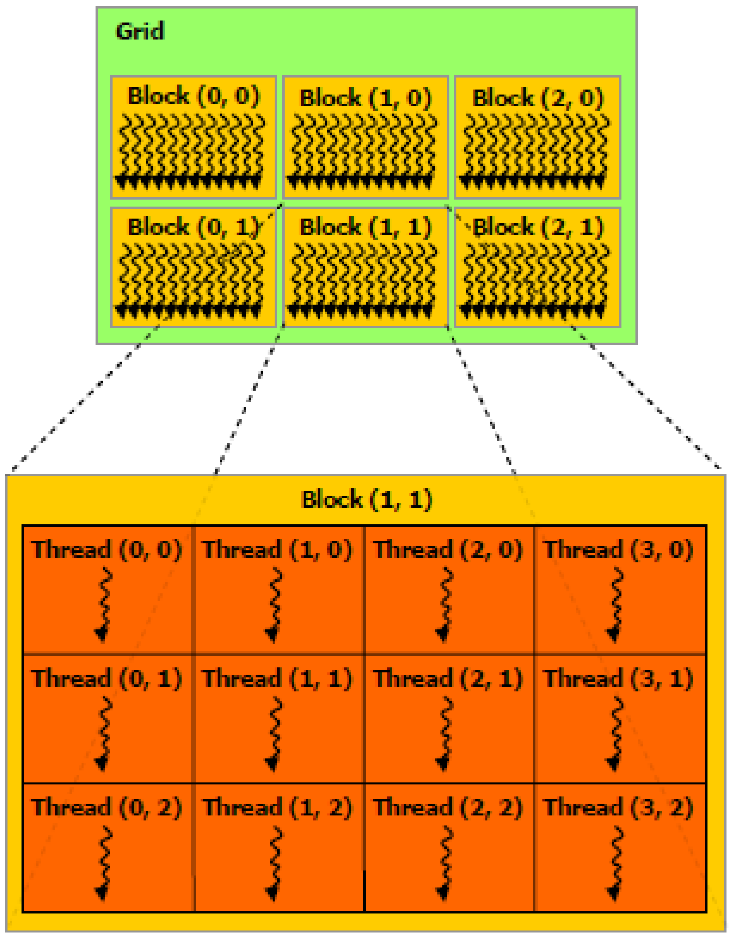

Figure 2-1. Grid of Thread Blocks

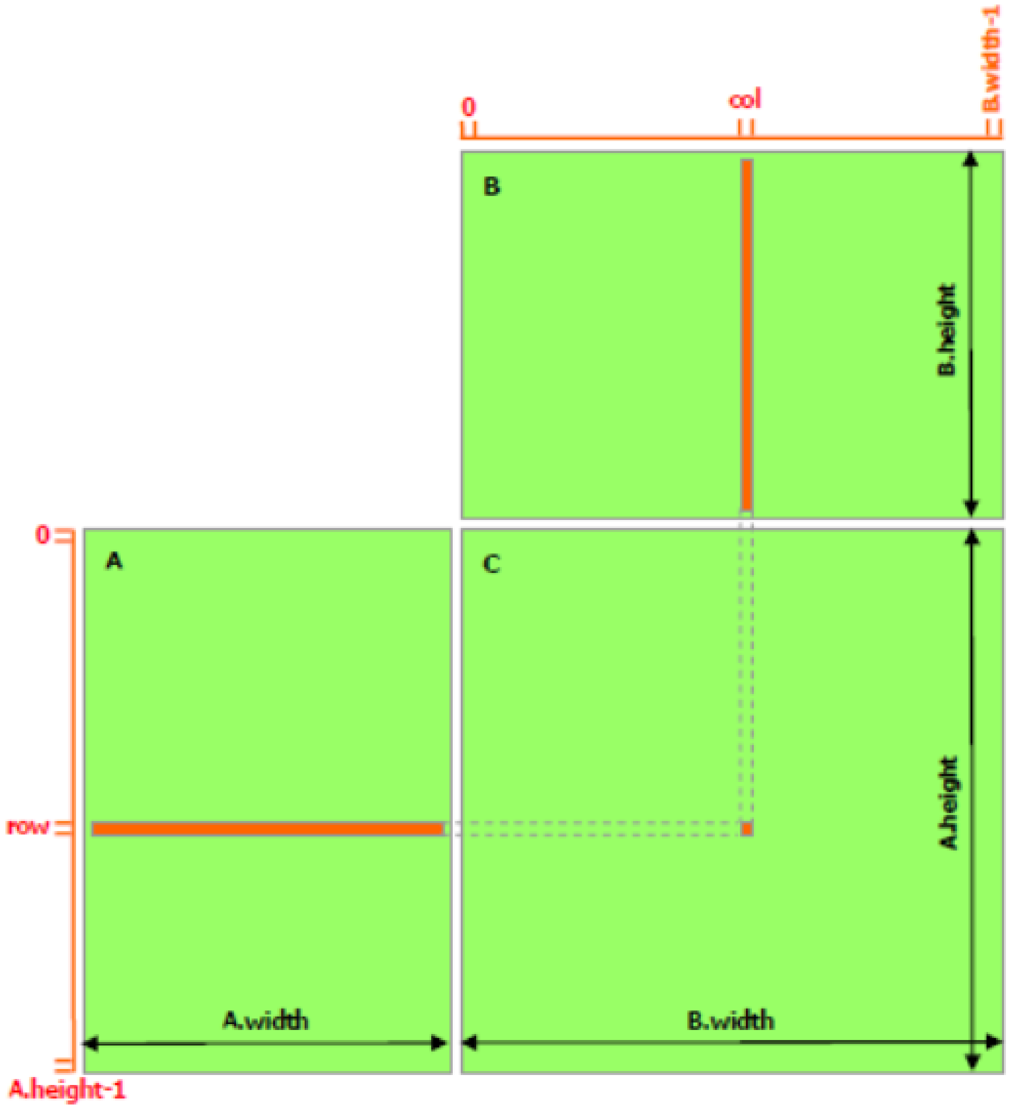

Figure 3-1. Matrix Multiplication without Shared Memory

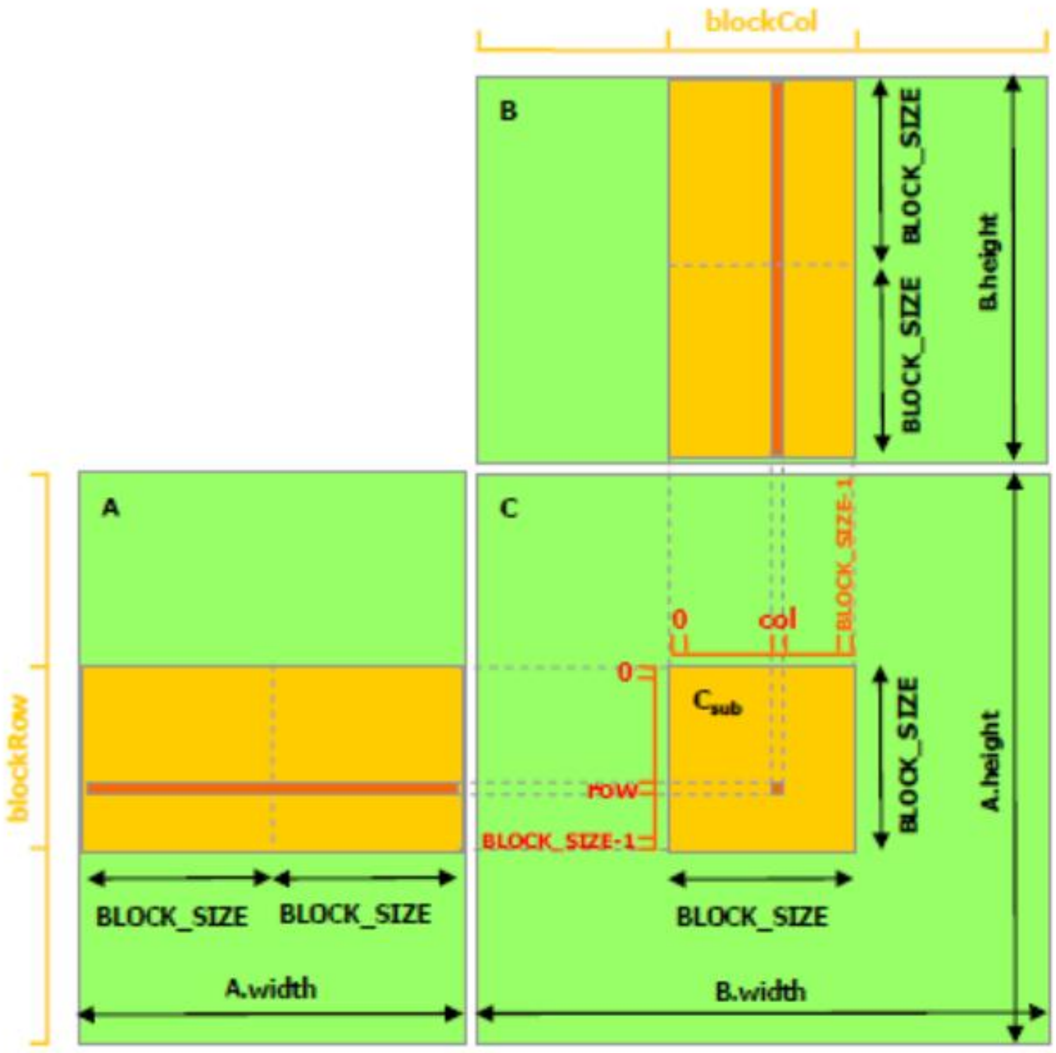

Figure 3-2. Matrix Multiplication with Shared Memory

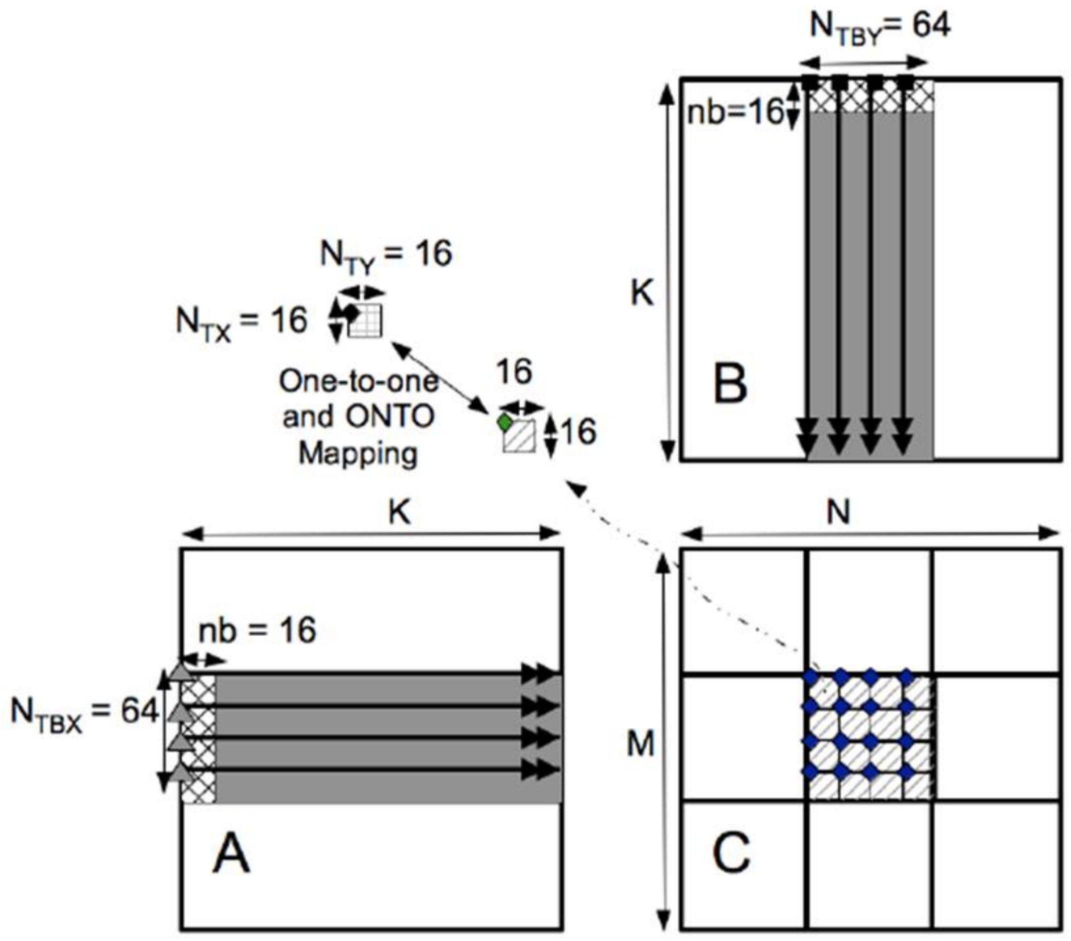

Fig. 2. The GPU GEMM ($C := \alpha AB + \beta C$) of a single TB for Fermi.

http://www.netlib.org/lapack/lawnspdf/lawn227.pdf