Tensor Core Tutorial

Made by Constant Park

sonicstage12@naver.com

2019-09-18

Reference

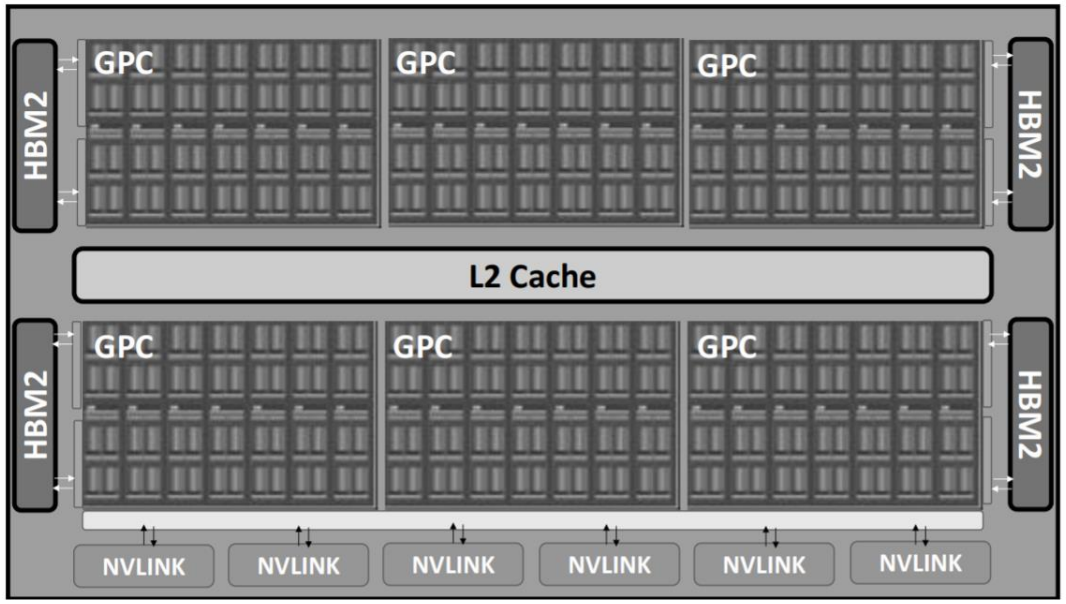
- Markidis, Stefano, et al. "Nvidia tensor core programmability, performance & precision." 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2018.
- Carrasco, Roberto, Raimundo Vega, and Cristóbal A. Navarro. "Analyzing GPU Tensor Core Potential for Fast Reductions." 2018 37th International Conference of the Chilean Computer Science Society (SCCC). IEEE, 2018.
- https://gigglehd.com/gg/mobile/1905675

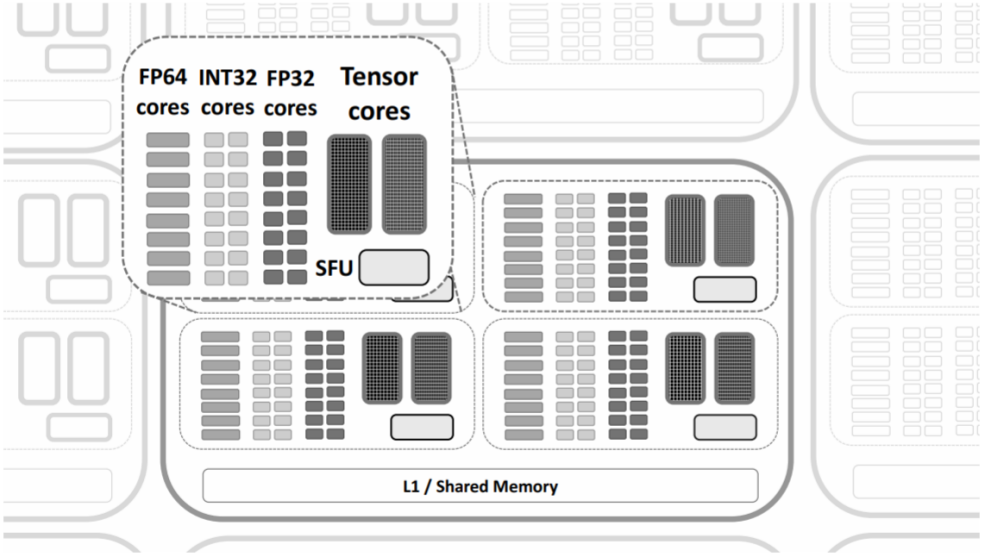
Tensor Core #1

- 튜링/볼타 GPU 아키텍처의 특징
  - 새로운 RT코어, 레이 트레이싱을 지원하는 AI 추론용 **텐서 코어**

Tensor Core #2

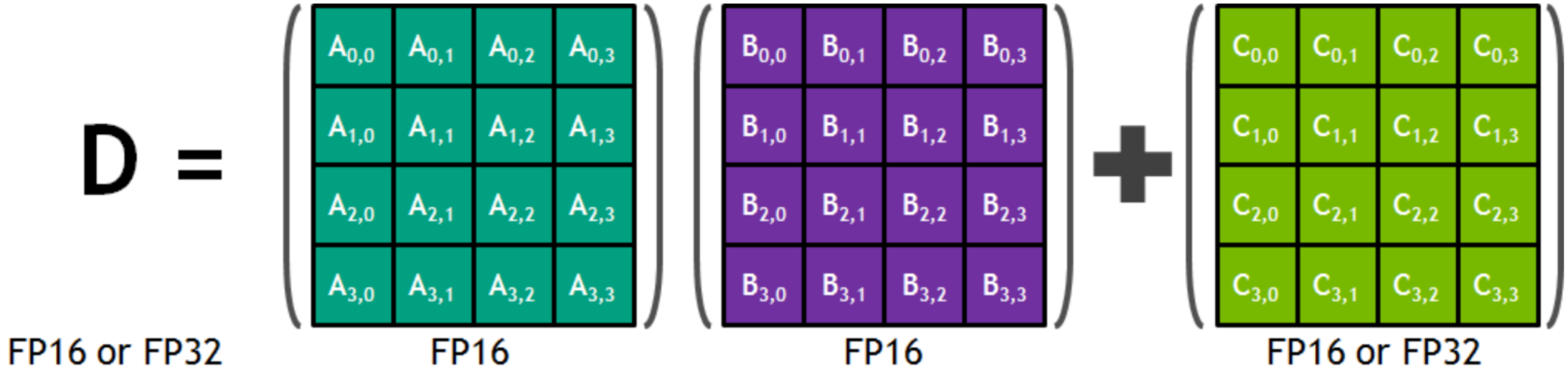
- Graphics Processing Clusters (GPC), HBM2, etc.
- **SM에 8개의 텐서 코어가 존재함**
- **8 (텐서 코어) * 64 (름**





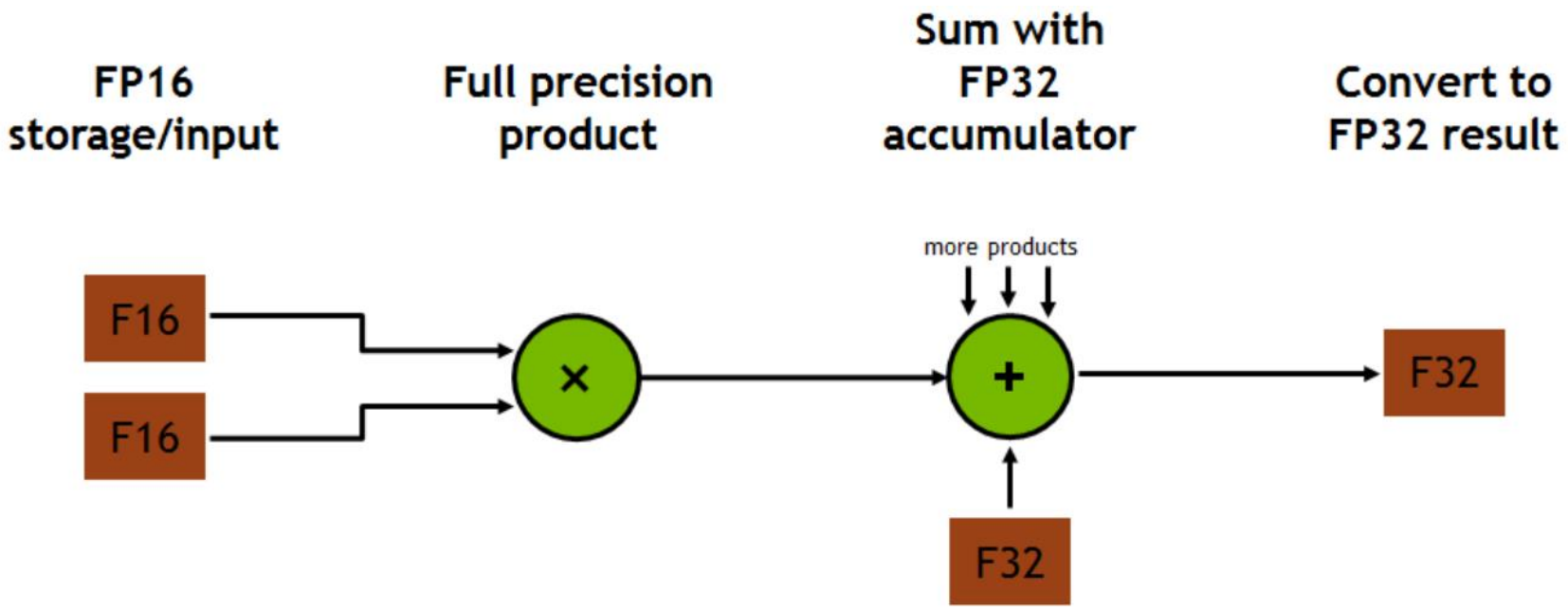
Tensor Core #3

- Designed for **4x4 matrix multiplication**



Tensor Core #4

- **FP16/FP32** computation



Tensor Core 정리

- 텐서 코어는 4x4 행렬의 곱셈만 계산 가능
- 행렬의 크기가 이보다 크다면 타일링을 사용하여 계산해야 됨
- cuBLAS, cuDNN, Warp-Level Matrix Operations API

cuBLAS

- **K, lda, ldb, ldc는 8의 배수, m은 4의 배수가 되어 함**
- CUDA_R_16F (Half), CUDA_R_32F (Full)

```
// Now using cuBLAS
printf("Running with cuBLAS...\n");
cudaErrCheck(cudaEventRecord(startcublas));
cublasErrCheck(cublasGemmEx(cublasHandle, CUBLAS_OP_N, CUBLAS_OP_N,
            MATRIX_M, MATRIX_N, MATRIX_K,
            &alpha,
            a_fp16, CUDA_R_16F, MATRIX_M,
            b_fp16, CUDA_R_16F, MATRIX_K,
            &beta,
            c_cublas, CUDA_R_32F, MATRIX_M,
            CUDA_R_32F, CUBLAS_GEMM_DFALT_TENSOR_OP));
cudaErrCheck(cudaEventRecord(stopcublas));
```

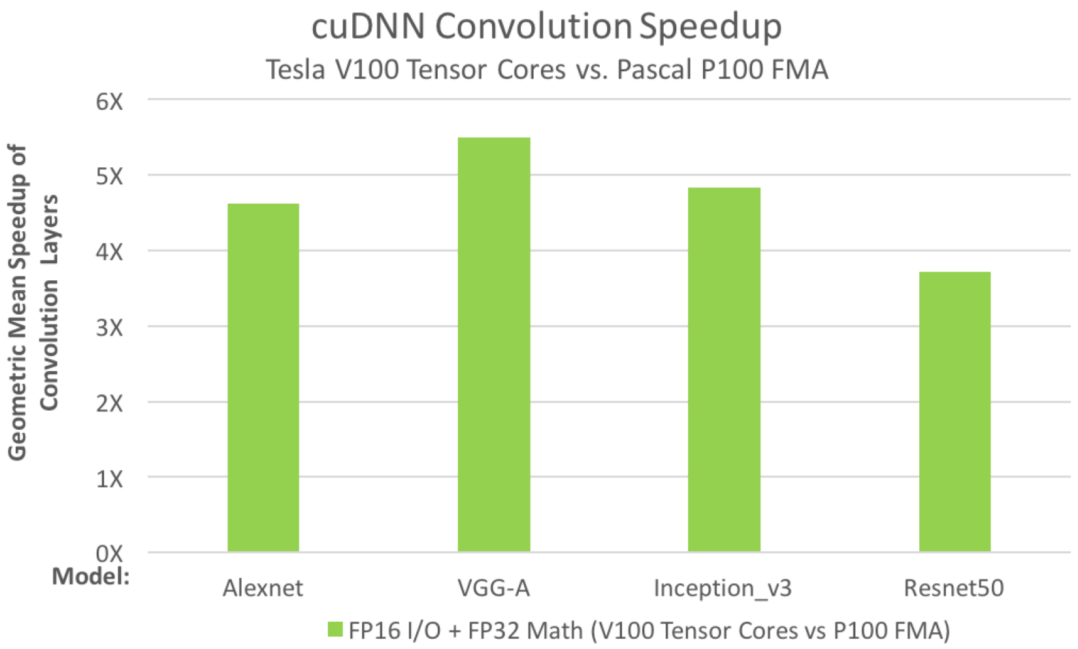
cuDNN

- **Math 타입은 CUDNN_TENSOR_OP_MATH이 되어야 함**
- **입력과 출력의 채널이 8의 배수가 되어야 동작**

```
// Set the math type to allow cuDNN to use Tensor Cores:
checkCudnnErr( cudnnSetConvolutionMathType(cudnnConvDesc, CUDNN_TENSOR_OP_MATH) );

// Choose a supported algorithm:
cudnnConvolutionFwdAlgo_t algo = CUDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM;

// Allocate your workspace:
checkCudnnErr( cudnnGetConvolutionForwardWorkspaceSize(handle_, cudnnIdesc,
                                                       cudnnFdesc, cudnnConvDesc,
                                                       cudnnOdesc, algo, &workSpaceSize) );
```



Programmatic Access to Tensor Cores

- 텐서 코어를 CUDA에서 프로그래밍 가능
- 행렬을 타일링해서 연산이 되도록 해야 함

```
__global__ void wmma_example(half *a, half *b, float *c, int M, int N, int K, float alpha, float beta) {
   // Leading dimensions. Packed with no transpositions.
   int lda = M;
   int ldb = K;
   int ldc = M;

   // Tile using a 2D grid
   int warpM = (blockIdx.x * blockDim.x + threadIdx.x) / warpSize;
   int warpN = (blockIdx.y * blockDim.y + threadIdx.y);

   // Declare the fragments
   wmma::fragment<wmma::matrix_a, WMMA_M, WMMA_N, WMMA_K, half, wmma::col_major> a_frag;
   wmma::fragment<wmma::matrix_b, WMMA_M, WMMA_N, WMMA_K, half, wmma::col_major> b_frag;
   wmma::fragment<wmma::accumulator, WMMA_M, WMMA_N, WMMA_K, float> acc_frag;
   wmma::fragment<wmma::accumulator, WMMA_M, WMMA_N, WMMA_K, float> c_frag;
```