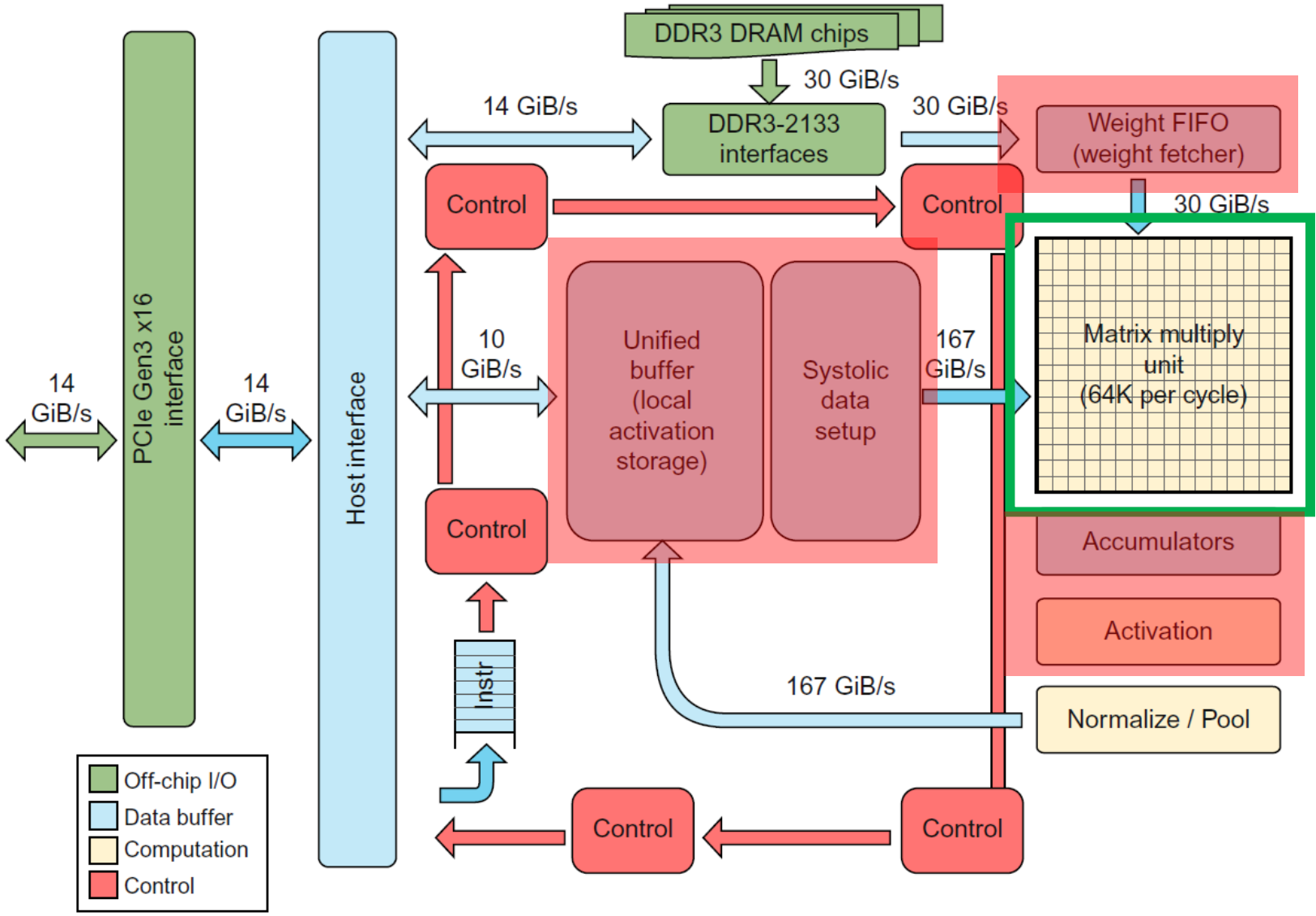
TPU Simulator

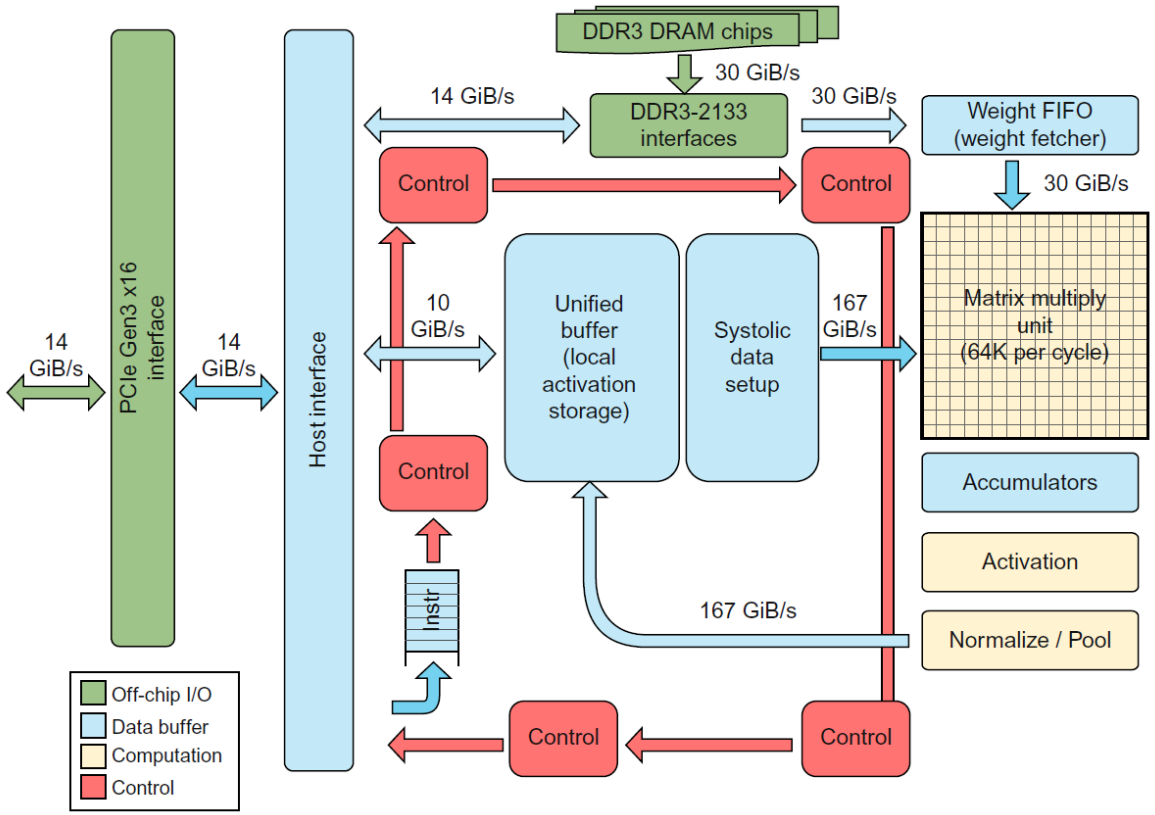
Dong Jae Lee

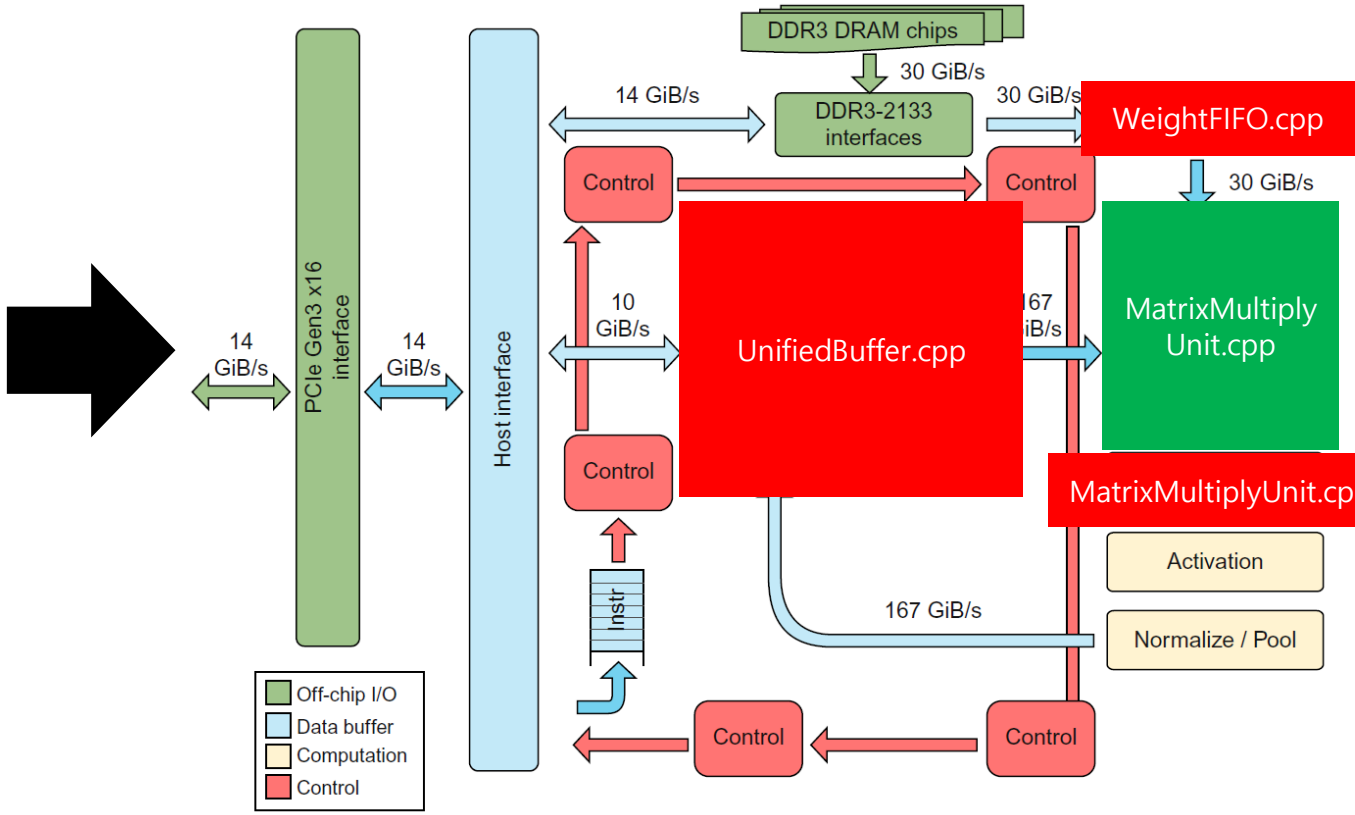
Assuming to be ideal

Focus of Simulation



Code Implementation





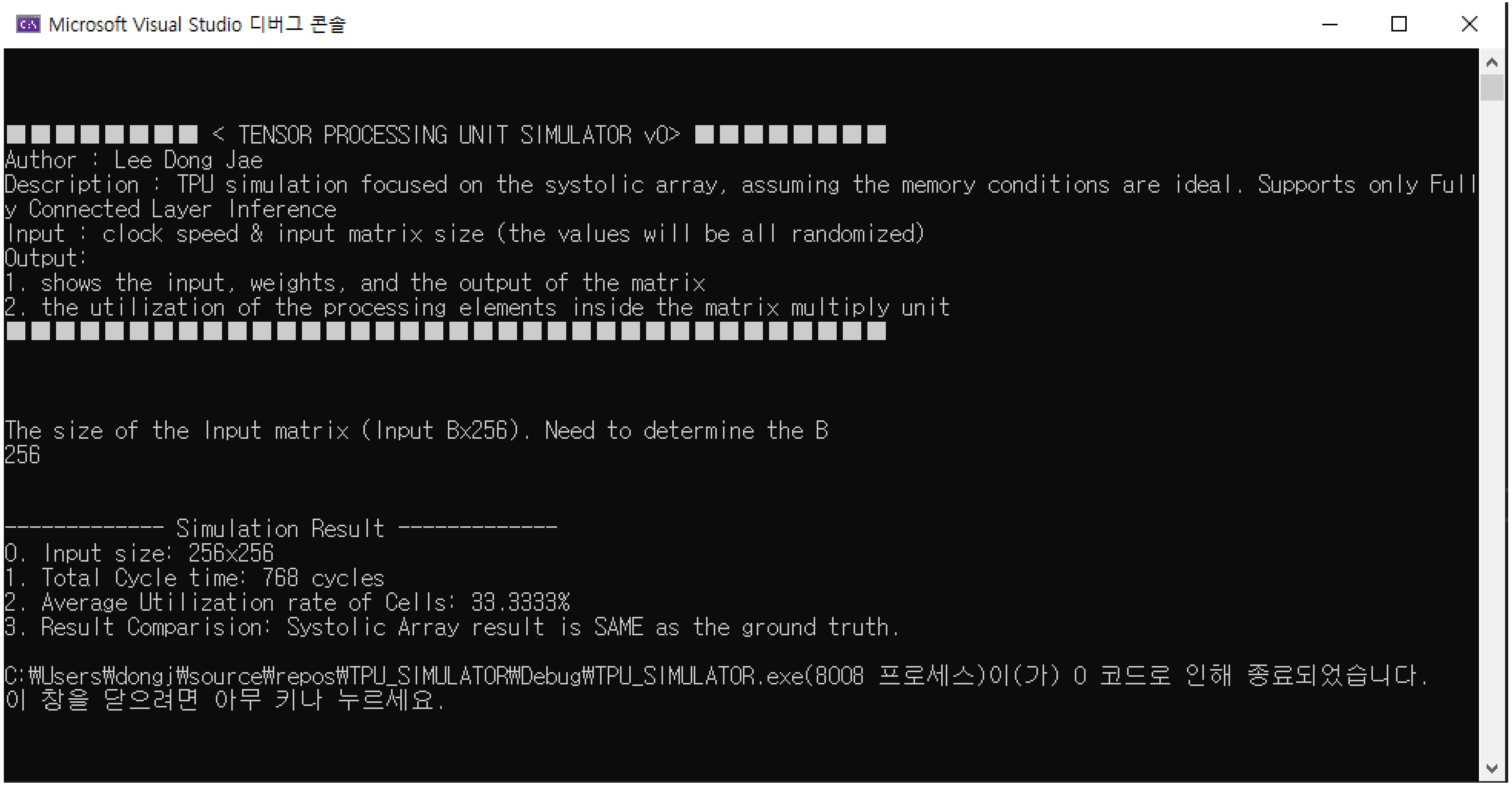
main.cpp

- 돌아가는 순서
- 1단계: 입력 행렬의 크기는 Bx256 (TPU ISA에서 아예 지정해뒀음)인데 사용자가 B 크기를 지정한다. (1부터 256까지 가능)
- 2단계: 시뮬레이션 세팅 초기화 과정
- 3단계: 사이클을 반복하면서 systolic array simulation이 끝날때까지 대기
- 4단계: 시뮬레이션 결과가 끝나면 runAnalysis() 함수가 시뮬레이션 결과 출력

시뮬레이션 결과

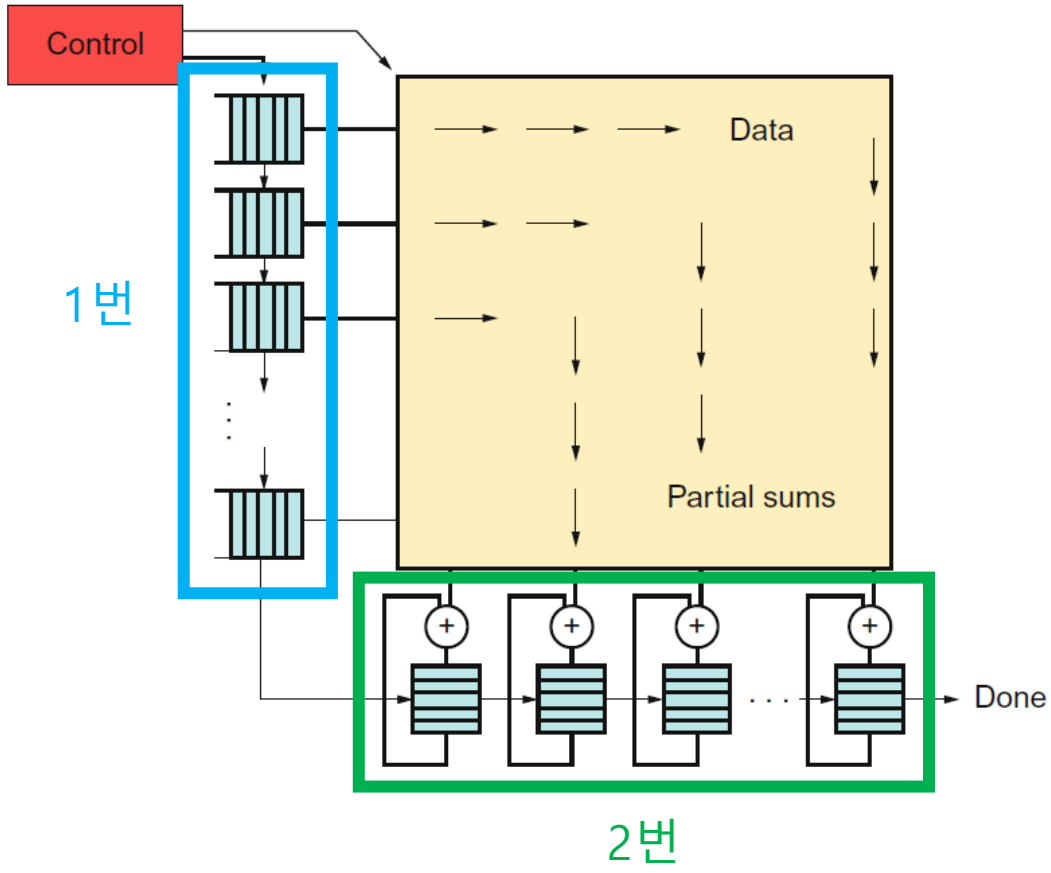
- 총 소요 사이클
- Systolic array 내 cell들의 평균 utilization 퍼센트 출력
- 실제 행렬 결과와 systolic array 결과 비교

Result of Simulation



MatrixMultiplyUnit.cpp - 변수

- __int8 InputBuffer[MATRIX_SIZE][MATRIX_SIZE * 2]: 1번
- __int32 Accumulator[ACCUMULATOR_SIZE][MATRIX_SIZE]: 2번
- __int32 buffer_accumulator[MATRIX_SIZE]: systolic array에서 accumulator로 넘어가기 전에 잠시 대기하는 공간 (accumulator에서 어떤 엔트리로 가야하는지 바로 알기가 어려워서)
- int accumulator_index[ACCUMULATOR_SIZE]: 4096개, 256개 32bit entry를 가진 accumulator에서 각각의 accumulator가 현재 비어있는 entry가 어디인지 저장해주는 저장소
- int ibuf_index: input buffer의 pop operation을 구현하기 위해 쓰임
- int count_idle_cell: 해당 systolic array 안에 몇개의 cell이 놀고 있는지 저장



MatrixMultiplyUnit.cpp – 함수 및 클래스

- **Class Cell**: Systolic Array 안에 들어가는 process element 클래스
  - void mac(): out = weight x in 진행
  - void propagate(): systolic array의 dataflow 진행 (오른쪽으로 input, 아래쪽으로 partial sum)
- **MMU_initialize()**: weight load, cell들 interconnect하기, accumulator 초기화
- **MMU_run()**: **한 클락 사이클이 돌아갈때** systolic array가 하는 작업들 (mac & systolic array의 dataflow 전개)

UnifiedBuffer.cpp

- std::vector<__int8> node_input: 입력 행렬의 element 저장
- void SystolicDataSetup(int mode, int input_length): 입력 행렬의 element를 -128 ~ 127의 범위에서 랜덤으로 생성한다. (TPU는 8bit integer로 양자화되어있으므로 동일하게 세팅을 맞췄음.)
- Memory는 ideal하다고 가정하므로, 구체적인 h/w적 구현은 생략

WeightFIFO.cpp

- __int8 preloadWeight(): weight 행렬의 element를 -128 ~ 127의 범위에서 랜덤으로 생성한다. (TPU는 8bit integer로 양자화되어있으므로 동일하게 세팅을 맞췄음.)
- Memory는 ideal하다고 가정하므로, 구체적인 h/w적 구현은 생략