AAHLS LabA report

R11943018 林子軒

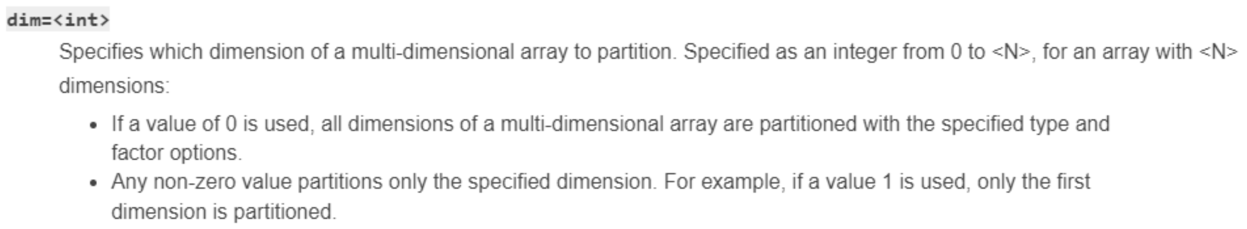
Github link: https://github.com/ezpzsyuan00/AAHLS_labB

Briefly introduction to the algorithm or overall system

This is an example of a systolic array applied to matrix multiplication.

Explain the original code/system/pragmas and how you implemented it

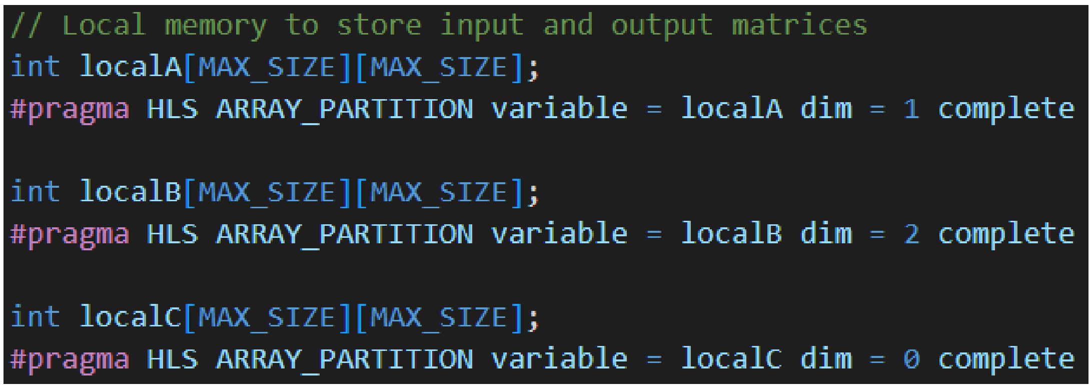
Vitis High-Level Synthesis User Guide:



`dim=<int>`

Specifies which dimension of a multi-dimensional array to partition. Specified as an integer from 0 to <N>, for an array with <N> dimensions:

- If a value of 0 is used, all dimensions of a multi-dimensional array are partitioned with the specified type and factor options.
- Any non-zero value partitions only the specified dimension. For example, if a value 1 is used, only the first dimension is partitioned.



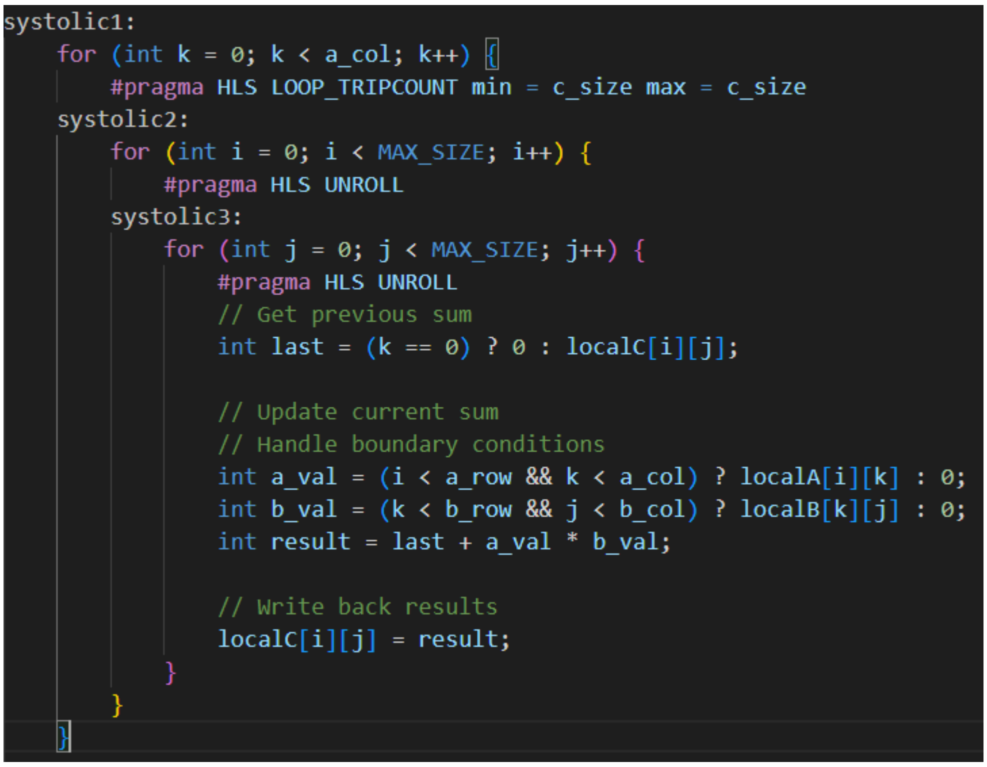
```
// Local memory to store input and output matrices
int localA[MAX_SIZE][MAX_SIZE];
#pragma HLS ARRAY_PARTITION variable = localA dim = 1 complete

int localB[MAX_SIZE][MAX_SIZE];
#pragma HLS ARRAY_PARTITION variable = localB dim = 2 complete

int localC[MAX_SIZE][MAX_SIZE];
#pragma HLS ARRAY_PARTITION variable = localC dim = 0 complete
```

The first pragma partitions the first dimension (rows) of matrix A into individual elements.

The second pragma partitions the second dimension (columns) of matrix B into individual elements.

The third pragma partitions all dimension of matrix C into individual elements.

Therefor, we can compute each element of the matrix C requires a row of matrix A and a column of matrix B.



```
systolic1:
    for (int k = 0; k < a_col; k++) {
        #pragma HLS LOOP_TRIPCOUNT min = c_size max = c_size
    systolic2:
        for (int i = 0; i < MAX_SIZE; i++) {
            #pragma HLS UNROLL
        systolic3:
            for (int j = 0; j < MAX_SIZE; j++) {
                #pragma HLS UNROLL
                // Get previous sum
                int last = (k == 0) ? 0 : localC[i][j];

                // Update current sum
                // Handle boundary conditions
                int a_val = (i < a_row && k < a_col) ? localA[i][k] : 0;
                int b_val = (k < b_row && j < b_col) ? localB[k][j] : 0;
                int result = last + a_val * b_val;

                // Write back results
                localC[i][j] = result;
            }
        }
    }
```

#pragma HLS UNROLL allows the loop to be fully or partially unrolled. Fully unrolling the loop creates a copy of the loop body for each loop iteration, so the whole loop can run at the same time Partially unrolling a loop can create N copies of the loop body and reduce the loop iterations.

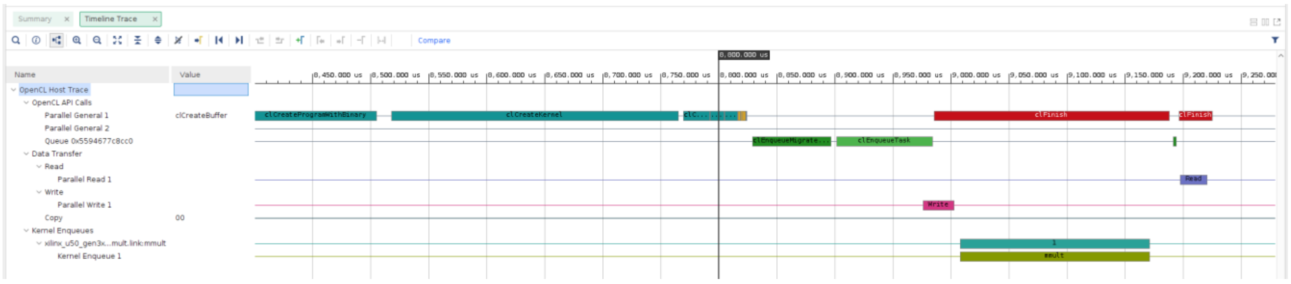
In cases of that the loop latency is unknown or cannot be calculated, ***#pragma HLS LOOP_TRIPCOUNT*** specifies the total number of iterations performed by a loop, which giving aid to the Vitis HLS tool to report the total latency of each loop in the reports. Because of that, it is for analysis only, and does not impact the results of synthesis.

Explain what you observed and learned

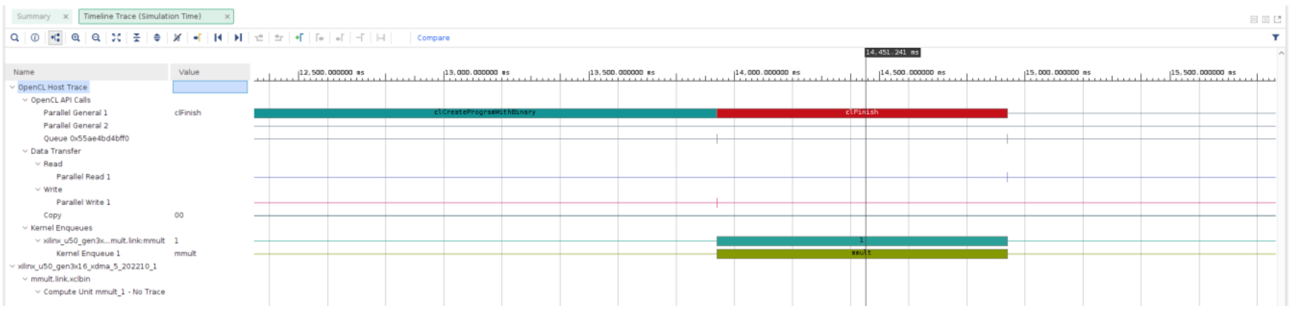
Vitis High-Level Synthesis User Guide (https://docs.xilinx.com/r/en-US/ug1399-vitis-hls/HLS-Pragmas) explains pragma very clearly. I think it will be very helpful for the final project.

Analyze the timing/performance/utilization

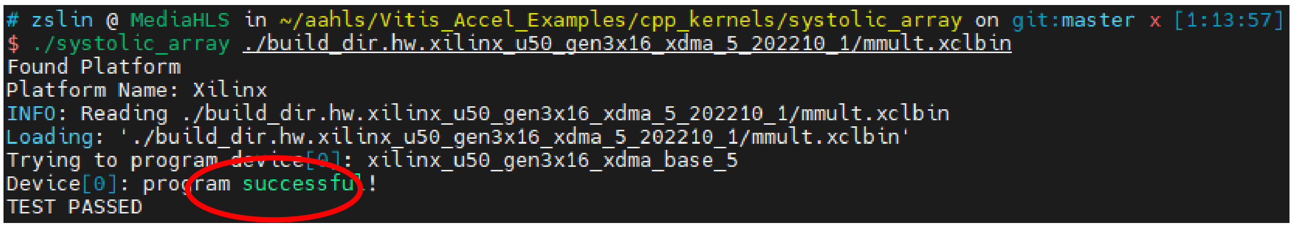
software emulation



hardware emulation

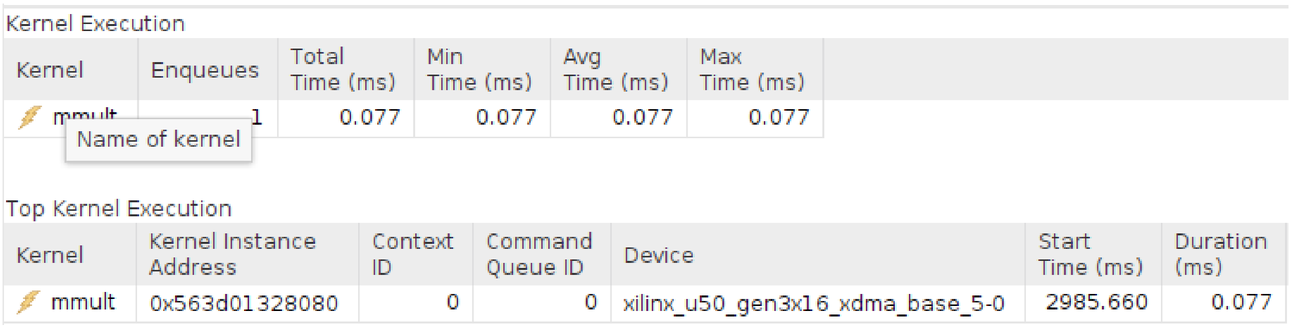


hardware



```
# zslin @ MediaHLS in ~/aahls/Vitis_Accel_Examples/cpp_kernels/systolic_array on git:master x [1:13:57]
$ ./systolic_array ./build_dir.hw.xilinx_u50_gen3x16_xdma_5_202210_1/mmult.xclbin
Found Platform
Platform Name: Xilinx
INFO: Reading ./build_dir.hw.xilinx_u50_gen3x16_xdma_5_202210_1/mmult.xclbin
Loading: './build_dir.hw.xilinx_u50_gen3x16_xdma_5_202210_1/mmult.xclbin'
Trying to program device[0]: xilinx_u50_gen3x16_xdma_base_5
Device[0]: program successful!
TEST PASSED
```

With ***#pragma HLS UNROLL***



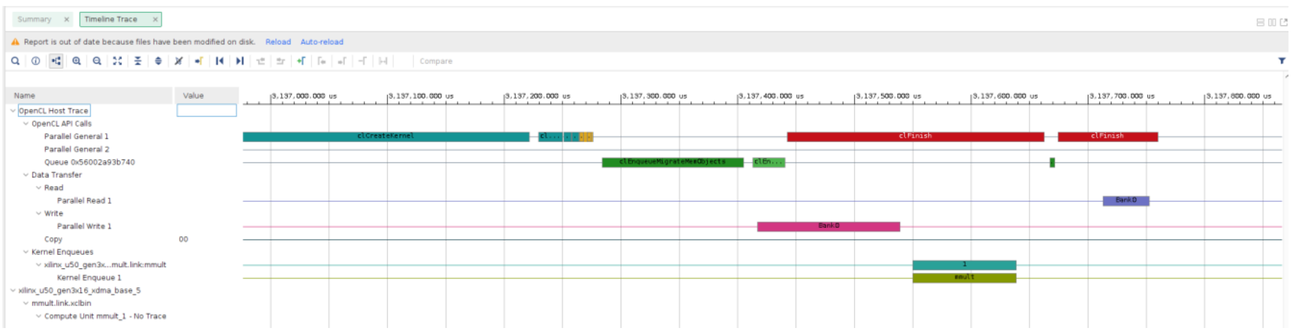
Kernel Execution

Kernel	Enqueues	Total Time (ms)	Min Time (ms)	Avg Time (ms)	Max Time (ms)
mmult	1	0.077	0.077	0.077	0.077

Name of kernel

Top Kernel Execution

Kernel	Kernel Instance Address	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)
mmult	0x563d01328080	0	0	xilinx_u50_gen3x16_xdma_base_5-0	2985.660	0.077



Explain what problem you encountered and how you solved it

This tutorial is written too briefly, so I have to go to the official website (https://xilinx.github.io/Vitis_Accel_Examples/2022.1/html/compile_execute.html) to check how to compilation and execution. But this is also an experience so that I can deal with problems more calmly in the future.