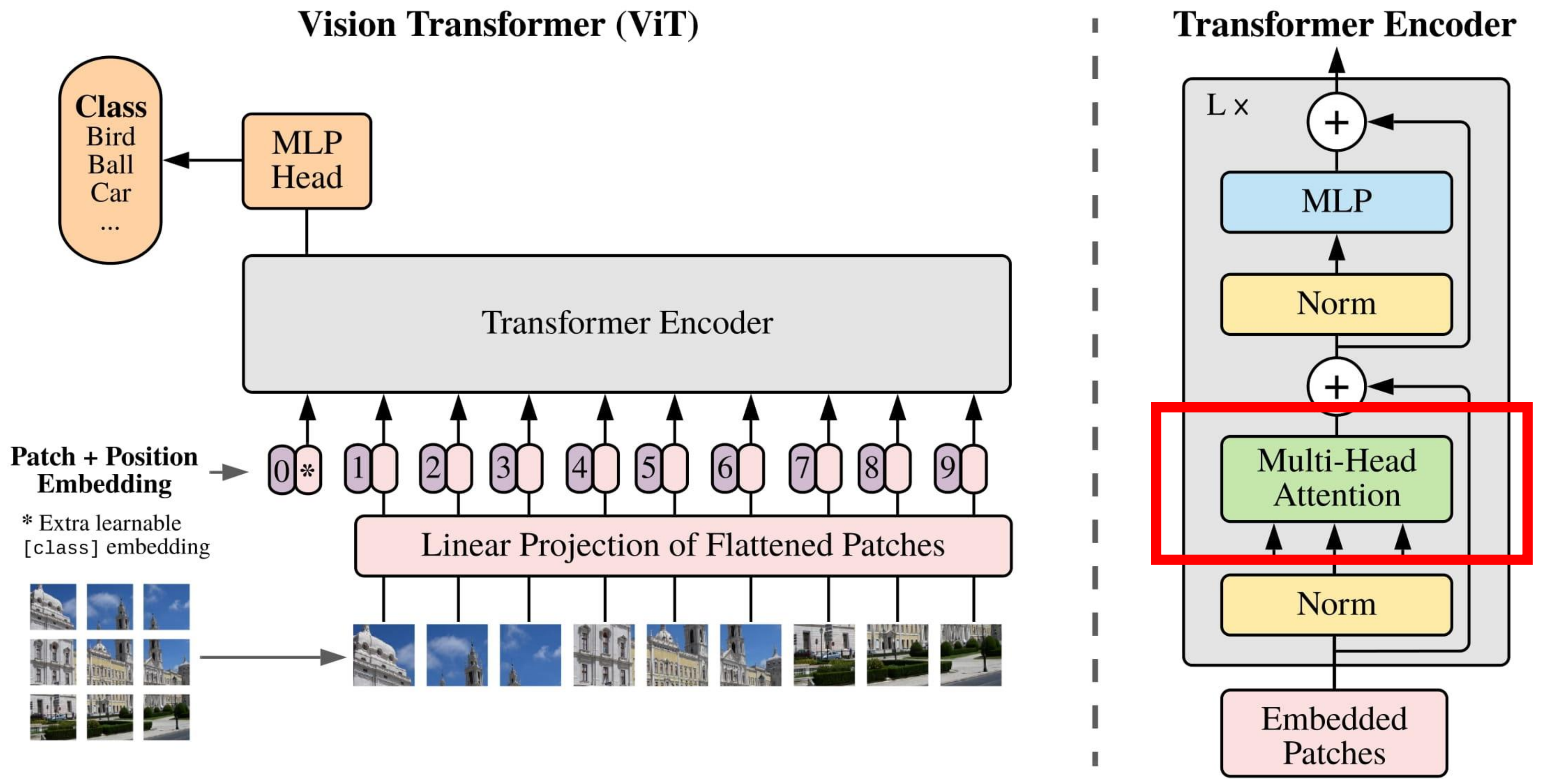
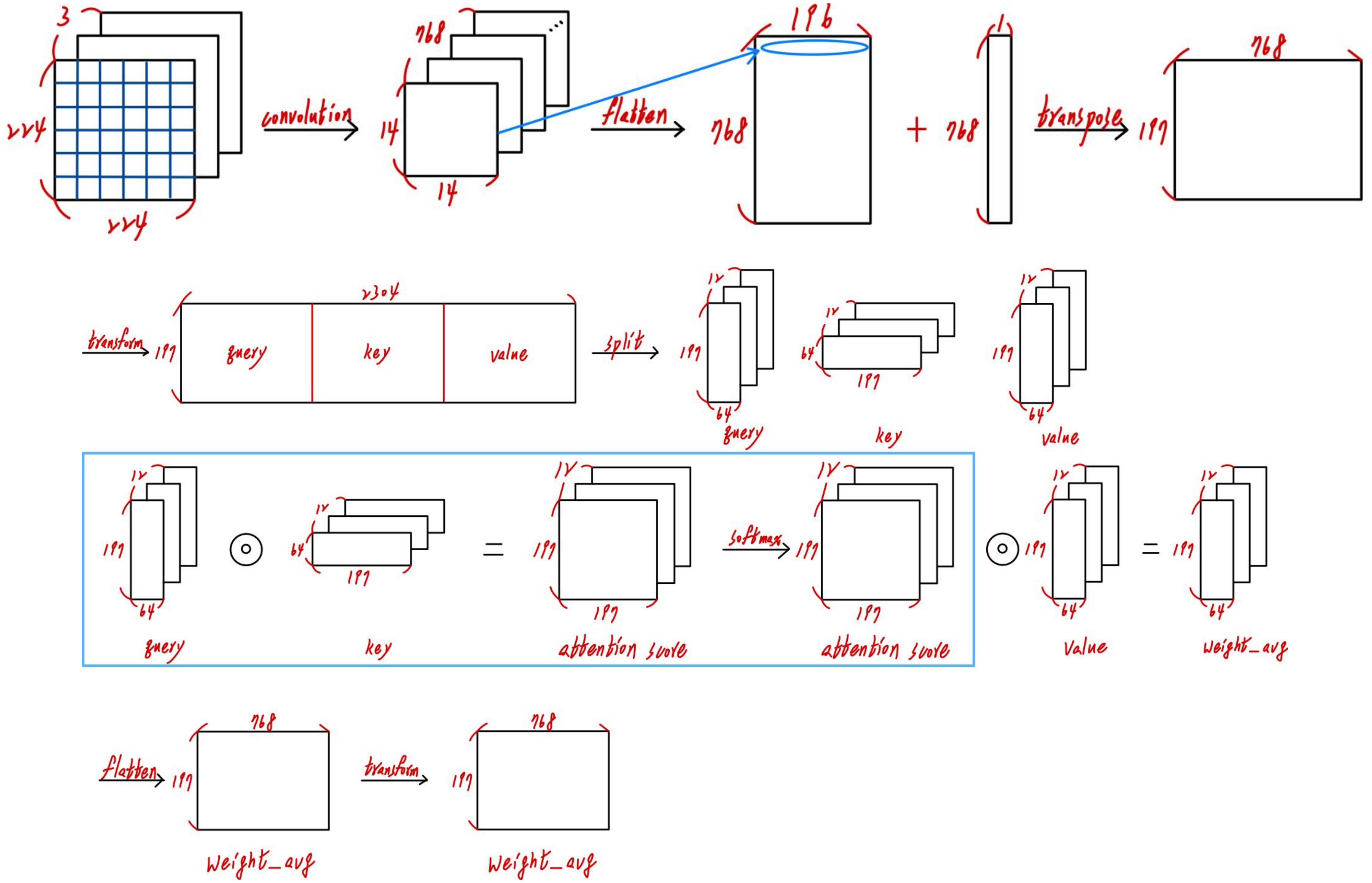
Vision Transformer

Application Acceleration with HLS

Architecture



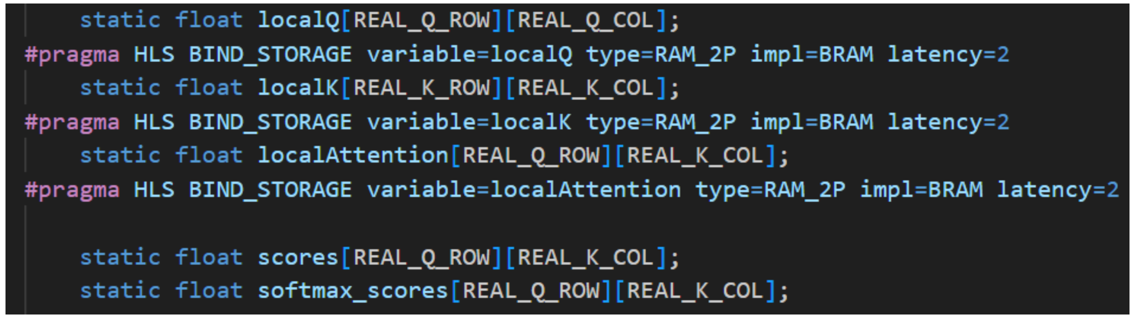
Hardware Input Dimension Derivation



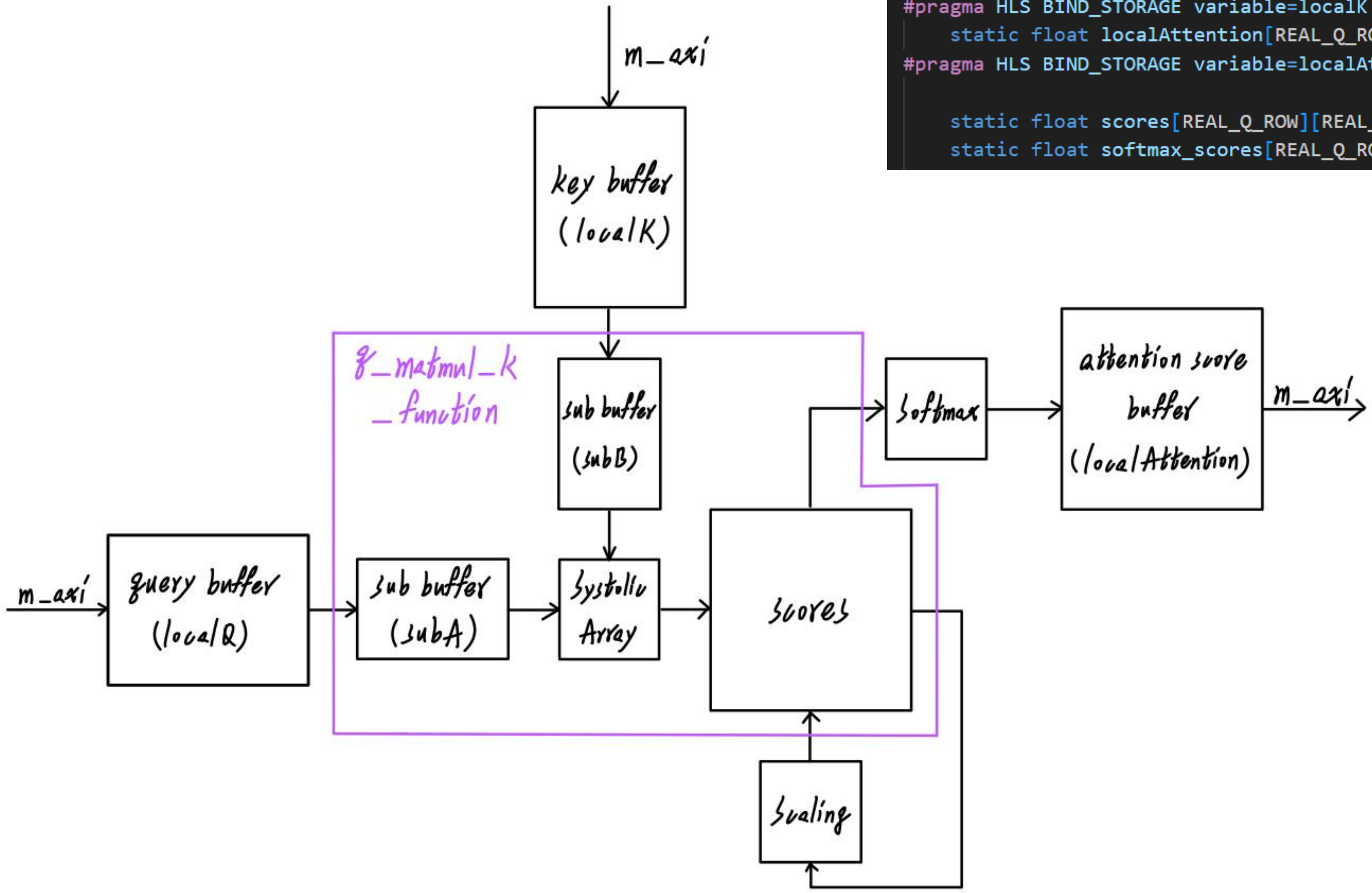
Patch Embedding

Multi-Head Attention

Block Diagram



```
    static float localQ[REAL_Q_ROW][REAL_Q_COL];
#pragma HLS BIND_STORAGE variable=localQ type=RAM_2P impl=BRAM latency=2
    static float localK[REAL_K_ROW][REAL_K_COL];
#pragma HLS BIND_STORAGE variable=localK type=RAM_2P impl=BRAM latency=2
    static float localAttention[REAL_Q_ROW][REAL_K_COL];
#pragma HLS BIND_STORAGE variable=localAttention type=RAM_2P impl=BRAM latency=2

    static float scores[REAL_Q_ROW][REAL_K_COL];
    static float softmax_scores[REAL_Q_ROW][REAL_K_COL];
```



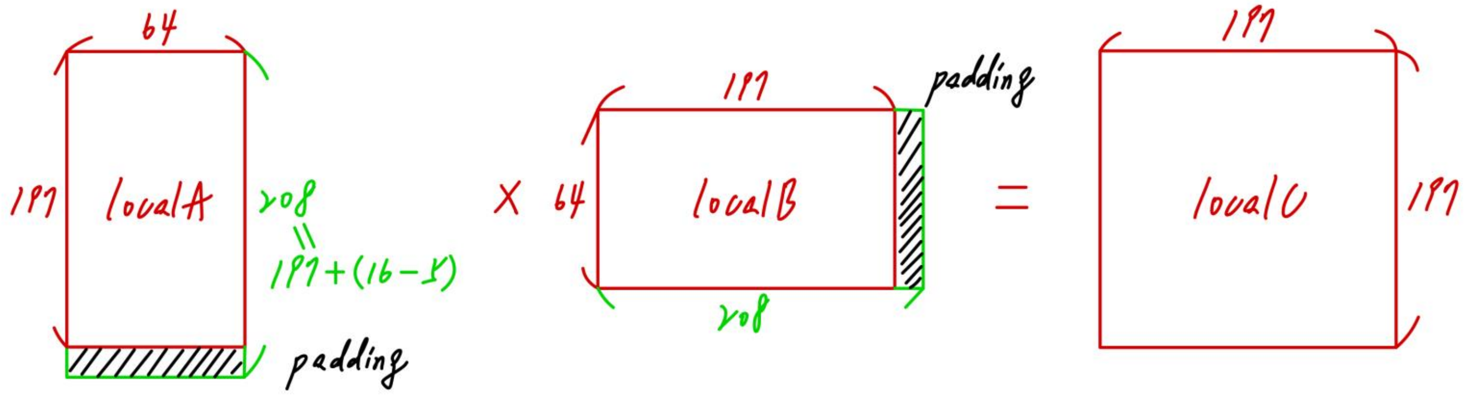
Hardware Implementation

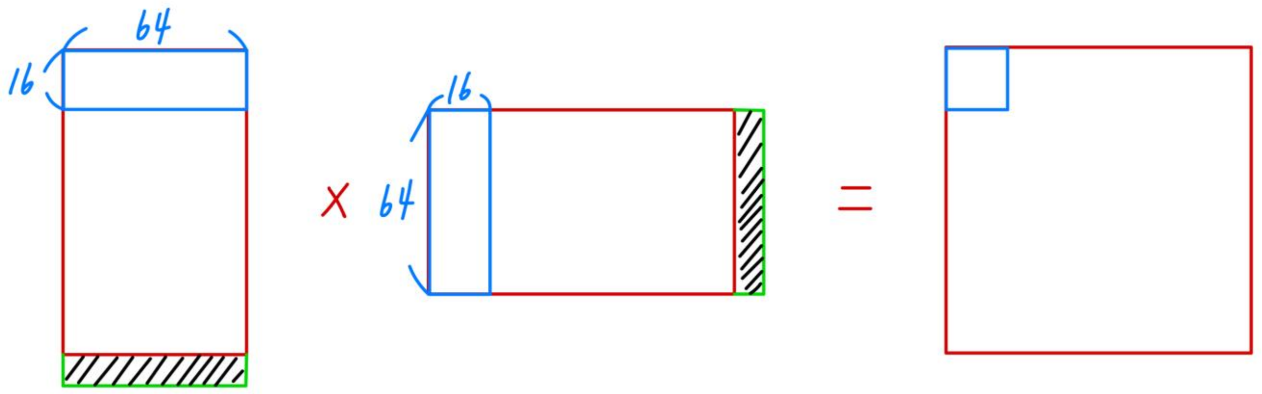
- Dot Product -> Systolic Array

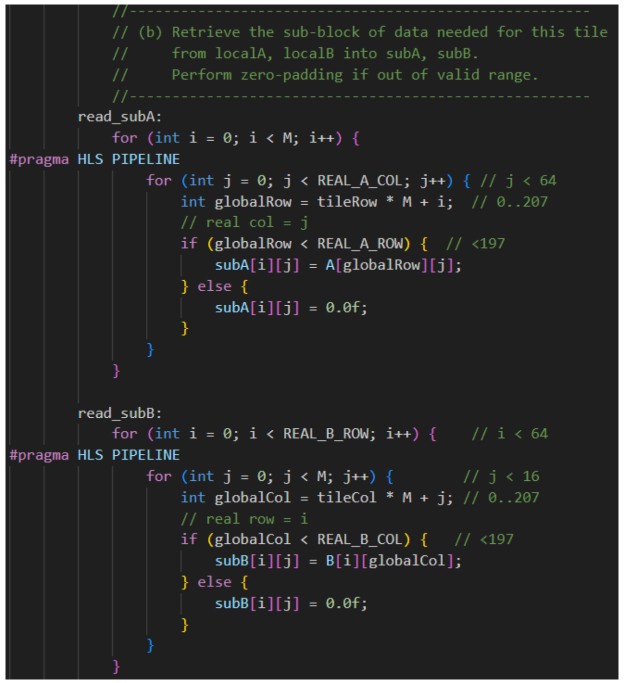
- Softmax Approximation

  exp(x) = 1 + x + (x * x) / 2 + (x * x * x) / 6

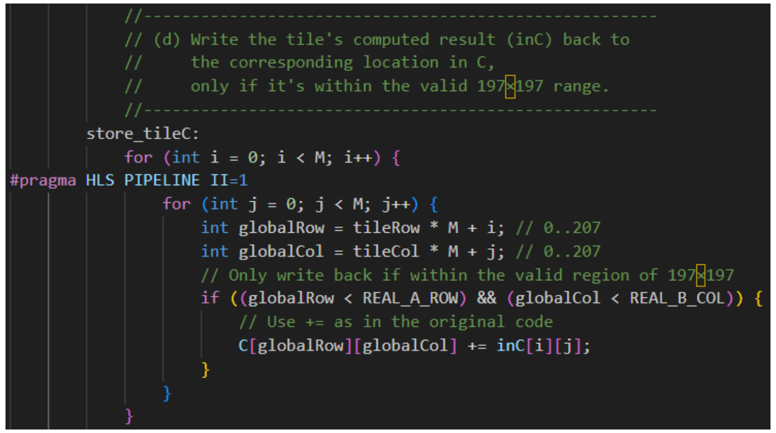
Systolic Array Design with tiling & padding





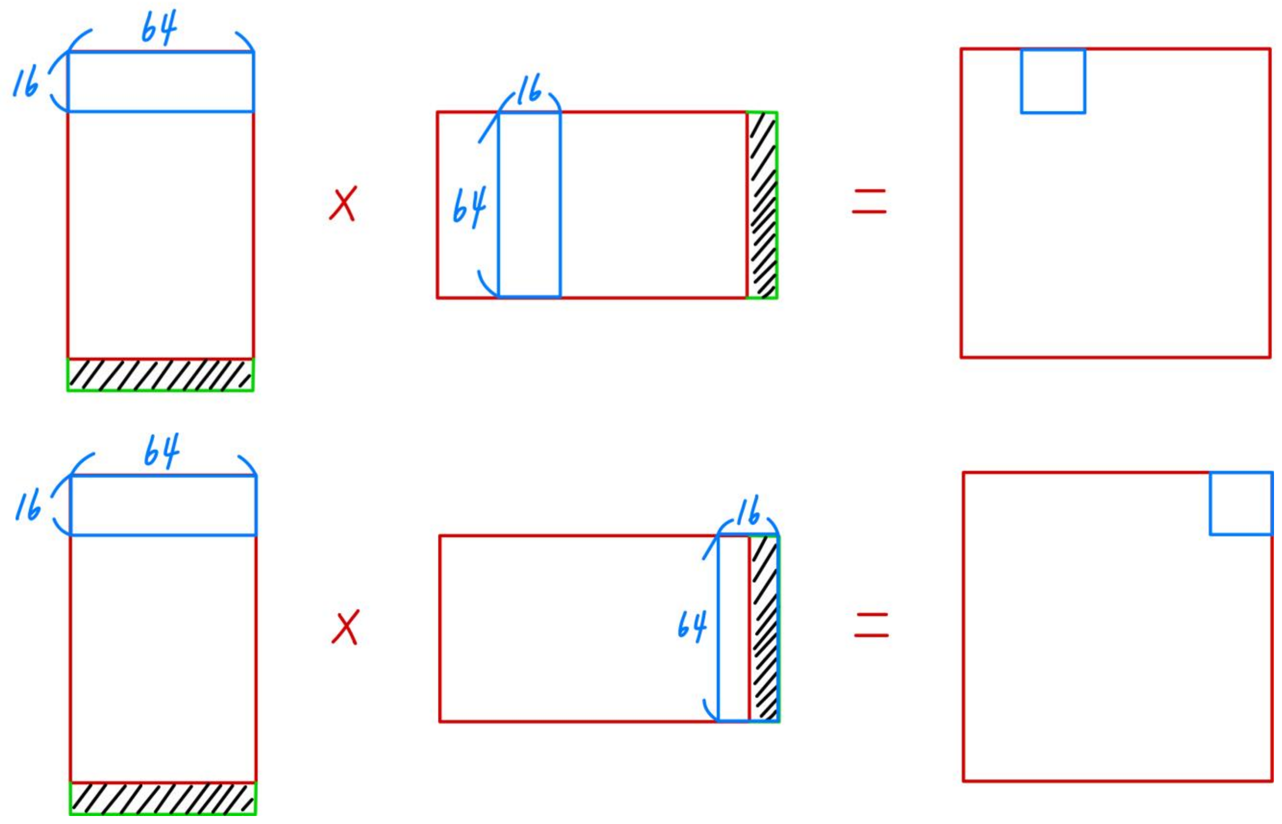


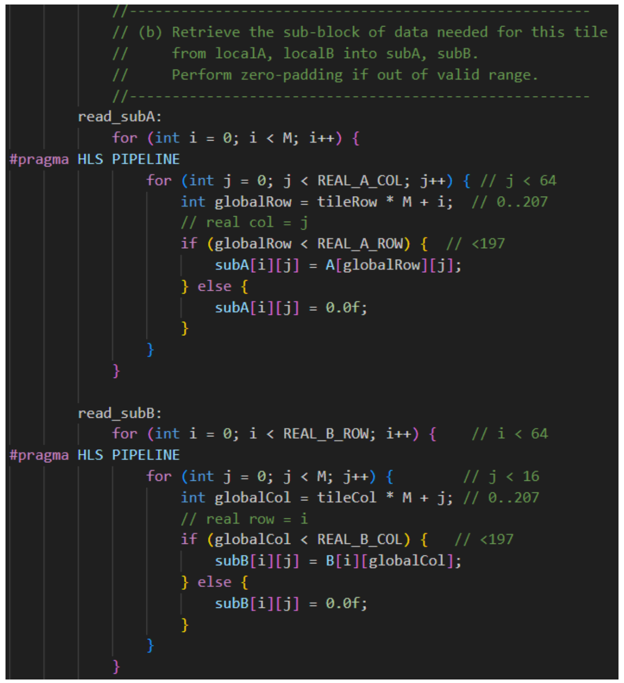
```
        //---------------------------------------------------------
        // (b) Retrieve the sub-block of data needed for this tile
        //     from localA, localB into subA, subB.
        //     Perform zero-padding if out of valid range.
        //---------------------------------------------------------
    read_subA:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE
            for (int j = 0; j < REAL_A_COL; j++) { // j < 64
                int globalRow = tileRow * M + i;  // 0..207
                // real col = j
                if (globalRow < REAL_A_ROW) {  // <197
                    subA[i][j] = A[globalRow][j];
                } else {
                    subA[i][j] = 0.0f;
                }
            }
        }

    read_subB:
        for (int i = 0; i < REAL_B_ROW; i++) {    // i < 64
#pragma HLS PIPELINE
            for (int j = 0; j < M; j++) {        // j < 16
                int globalCol = tileCol * M + j; // 0..207
                // real row = i
                if (globalCol < REAL_B_COL) {   // <197
                    subB[i][j] = B[i][globalCol];
                } else {
                    subB[i][j] = 0.0f;
                }
            }
        }
```



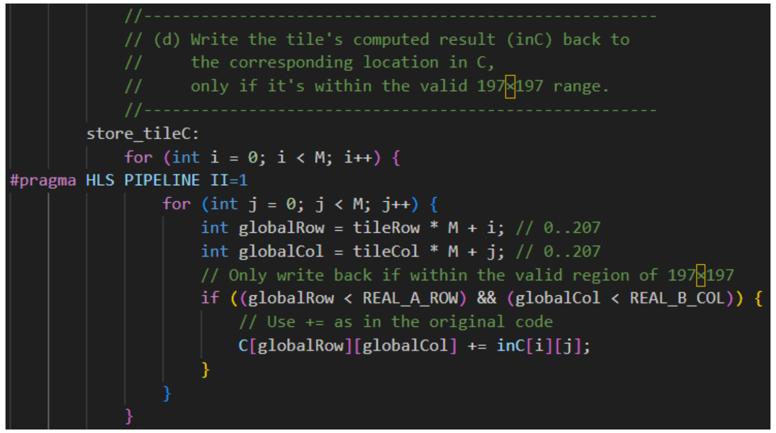
```
        //---------------------------------------------------------
        // (d) Write the tile's computed result (inC) back to
        //     the corresponding location in C,
        //     only if it's within the valid 197×197 range.
        //---------------------------------------------------------
    store_tileC:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE II=1
            for (int j = 0; j < M; j++) {
                int globalRow = tileRow * M + i; // 0..207
                int globalCol = tileCol * M + j; // 0..207
                // Only write back if within the valid region of 197×197
                if ((globalRow < REAL_A_ROW) && (globalCol < REAL_B_COL)) {
                    // Use += as in the original code
                    C[globalRow][globalCol] += inC[i][j];
                }
            }
        }
```

Systolic Array Design with tiling & padding



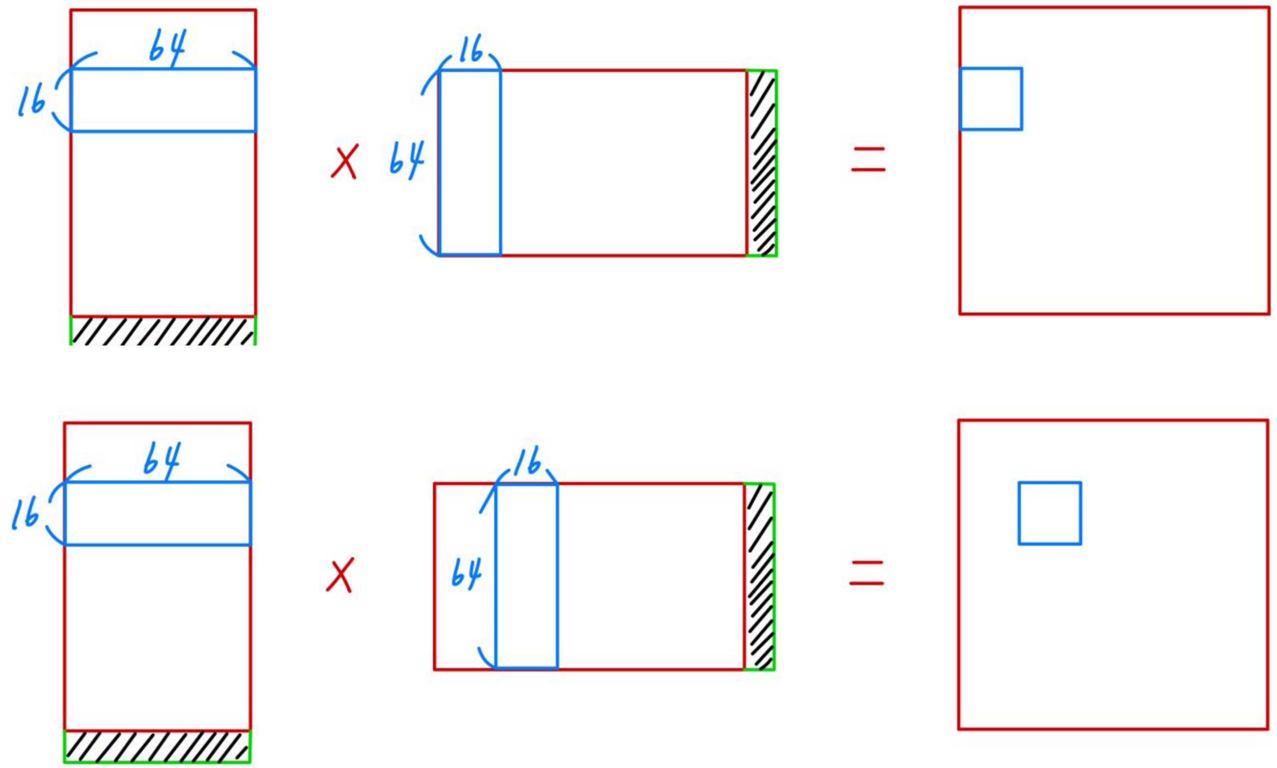


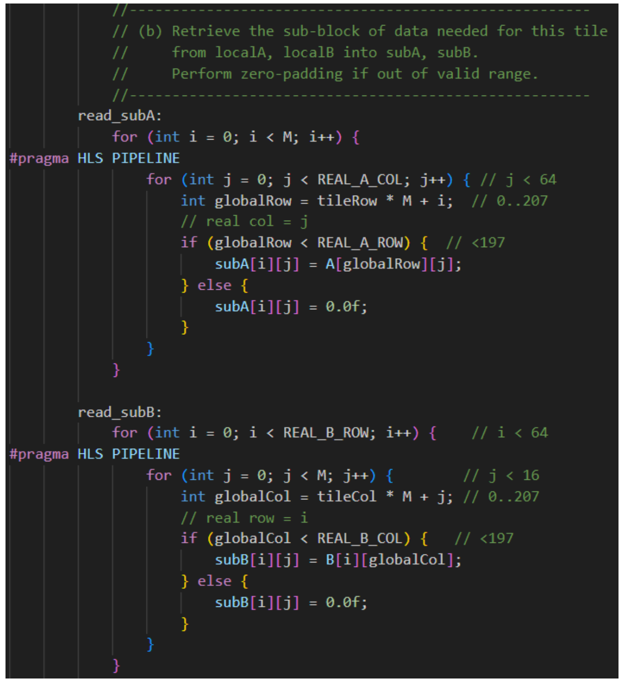
```
        //----------------------------------------------------
        // (b) Retrieve the sub-block of data needed for this tile
        //     from localA, localB into subA, subB.
        //     Perform zero-padding if out of valid range.
        //----------------------------------------------------
    read_subA:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE
            for (int j = 0; j < REAL_A_COL; j++) { // j < 64
                int globalRow = tileRow * M + i;  // 0..207
                // real col = j
                if (globalRow < REAL_A_ROW) {  // <197
                    subA[i][j] = A[globalRow][j];
                } else {
                    subA[i][j] = 0.0f;
                }
            }
        }

    read_subB:
        for (int i = 0; i < REAL_B_ROW; i++) {    // i < 64
#pragma HLS PIPELINE
            for (int j = 0; j < M; j++) {        // j < 16
                int globalCol = tileCol * M + j; // 0..207
                // real row = i
                if (globalCol < REAL_B_COL) {   // <197
                    subB[i][j] = B[i][globalCol];
                } else {
                    subB[i][j] = 0.0f;
                }
            }
        }
```



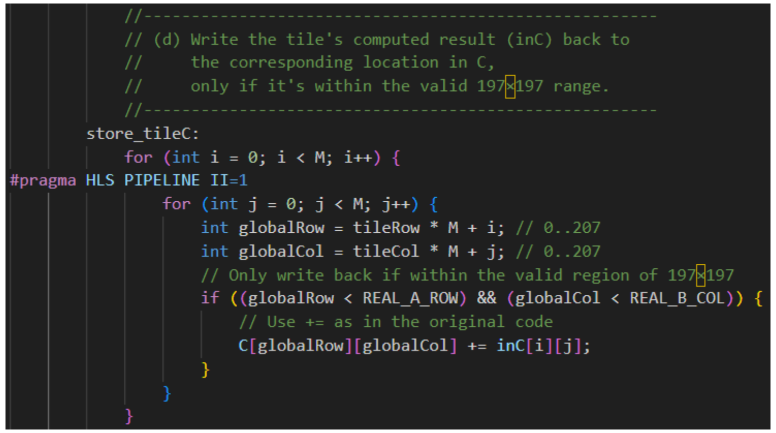
```
        //----------------------------------------------------
        // (d) Write the tile's computed result (inC) back to
        //     the corresponding location in C,
        //     only if it's within the valid 197×197 range.
        //----------------------------------------------------
    store_tileC:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE II=1
            for (int j = 0; j < M; j++) {
                int globalRow = tileRow * M + i; // 0..207
                int globalCol = tileCol * M + j; // 0..207
                // Only write back if within the valid region of 197×197
                if ((globalRow < REAL_A_ROW) && (globalCol < REAL_B_COL)) {
                    // Use += as in the original code
                    C[globalRow][globalCol] += inC[i][j];
                }
            }
        }
```

Systolic Array Design with tiling & padding



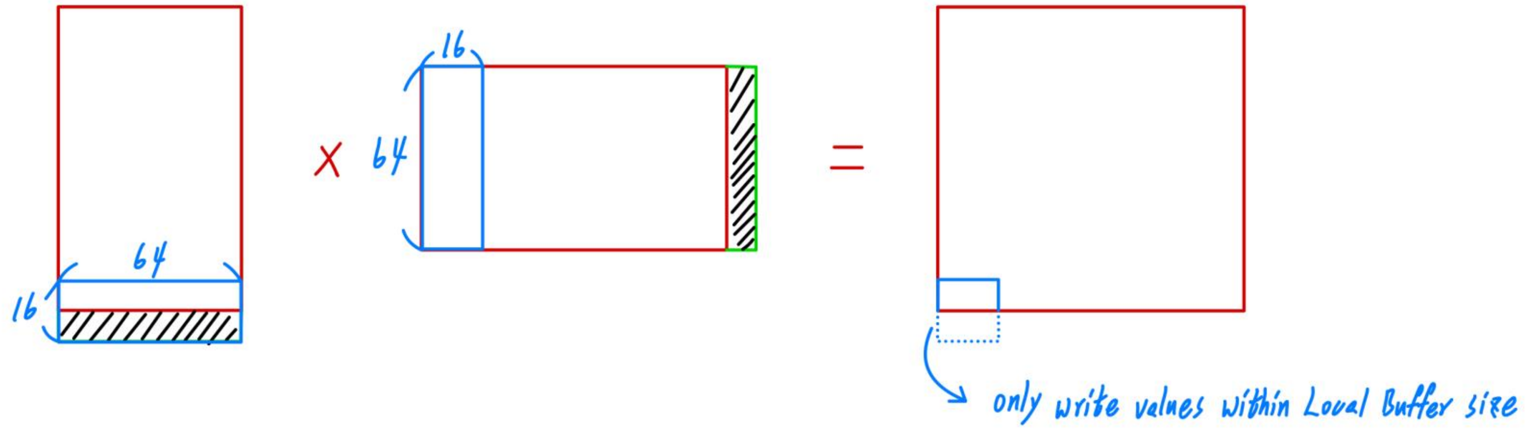


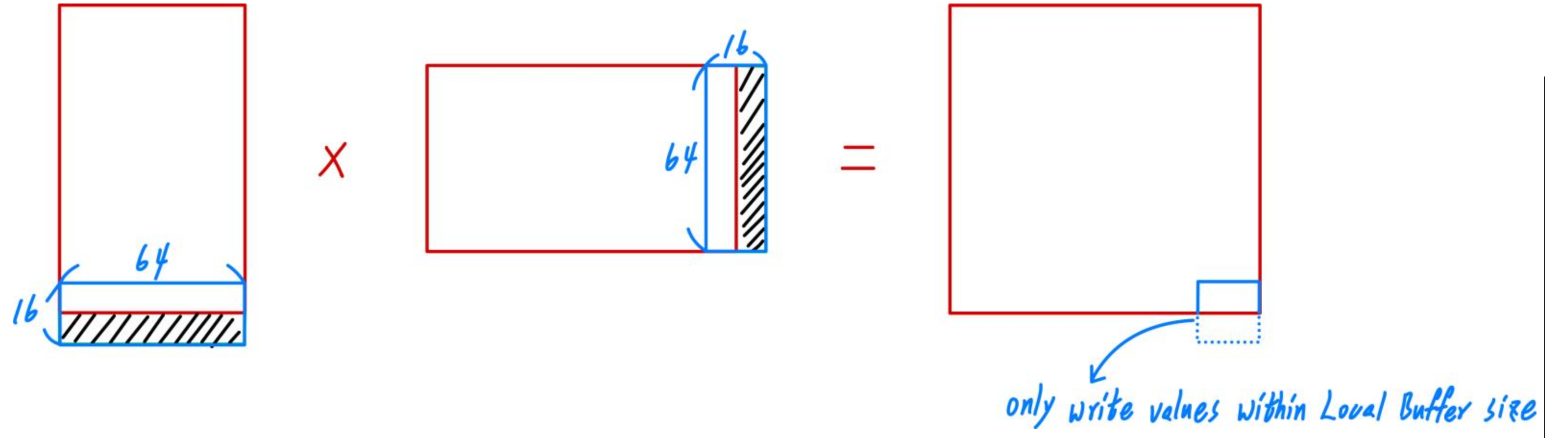
```
        //-------------------------------------------------------
        // (b) Retrieve the sub-block of data needed for this tile
        //     from localA, localB into subA, subB.
        //     Perform zero-padding if out of valid range.
        //-------------------------------------------------------
    read_subA:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE
            for (int j = 0; j < REAL_A_COL; j++) { // j < 64
                int globalRow = tileRow * M + i;  // 0..207
                // real col = j
                if (globalRow < REAL_A_ROW) {  // <197
                    subA[i][j] = A[globalRow][j];
                } else {
                    subA[i][j] = 0.0f;
                }
            }
        }

    read_subB:
        for (int i = 0; i < REAL_B_ROW; i++) {    // i < 64
#pragma HLS PIPELINE
            for (int j = 0; j < M; j++) {        // j < 16
                int globalCol = tileCol * M + j; // 0..207
                // real row = i
                if (globalCol < REAL_B_COL) {   // <197
                    subB[i][j] = B[i][globalCol];
                } else {
                    subB[i][j] = 0.0f;
                }
            }
        }
```

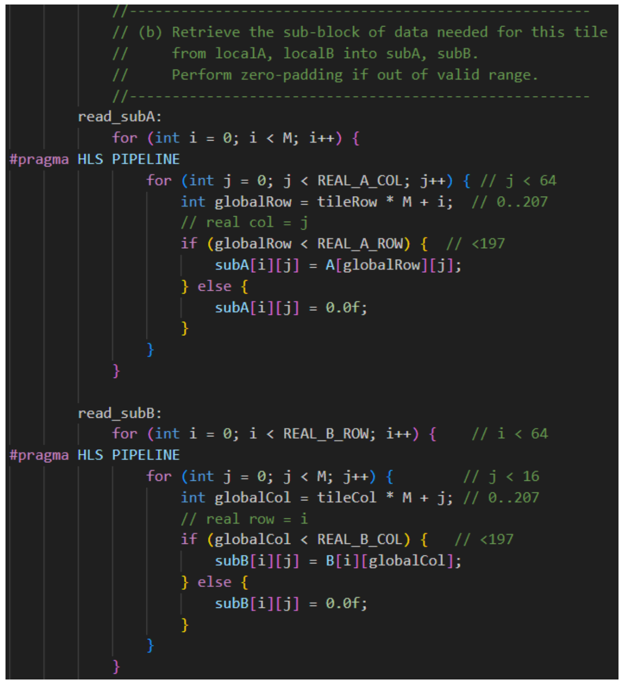


```
        //-------------------------------------------------------
        // (d) Write the tile's computed result (inC) back to
        //     the corresponding location in C,
        //     only if it's within the valid 197×197 range.
        //-------------------------------------------------------
    store_tileC:
        for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE II=1
            for (int j = 0; j < M; j++) {
                int globalRow = tileRow * M + i; // 0..207
                int globalCol = tileCol * M + j; // 0..207
                // Only write back if within the valid region of 197×197
                if ((globalRow < REAL_A_ROW) && (globalCol < REAL_B_COL)) {
                    // Use += as in the original code
                    C[globalRow][globalCol] += inC[i][j];
                }
            }
        }
```

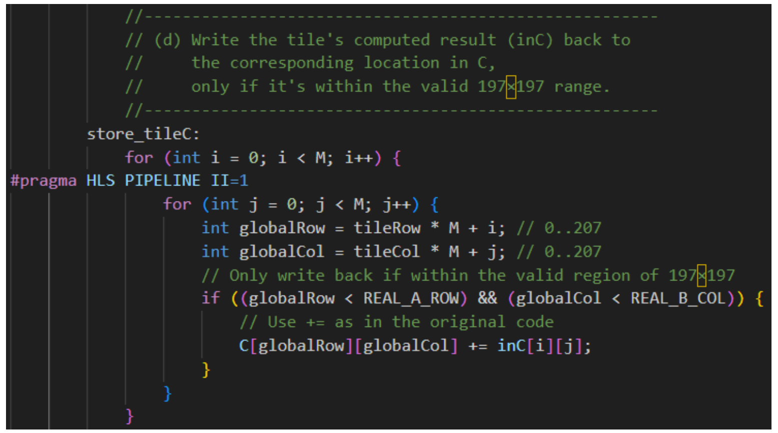
Systolic Array Design with tiling & padding





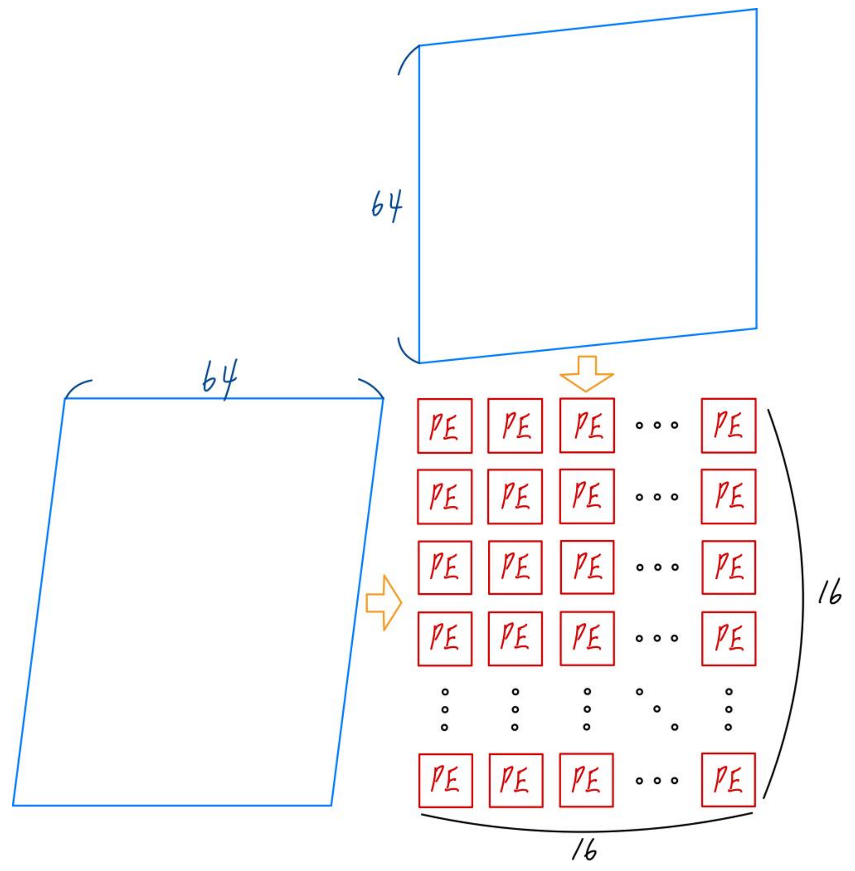


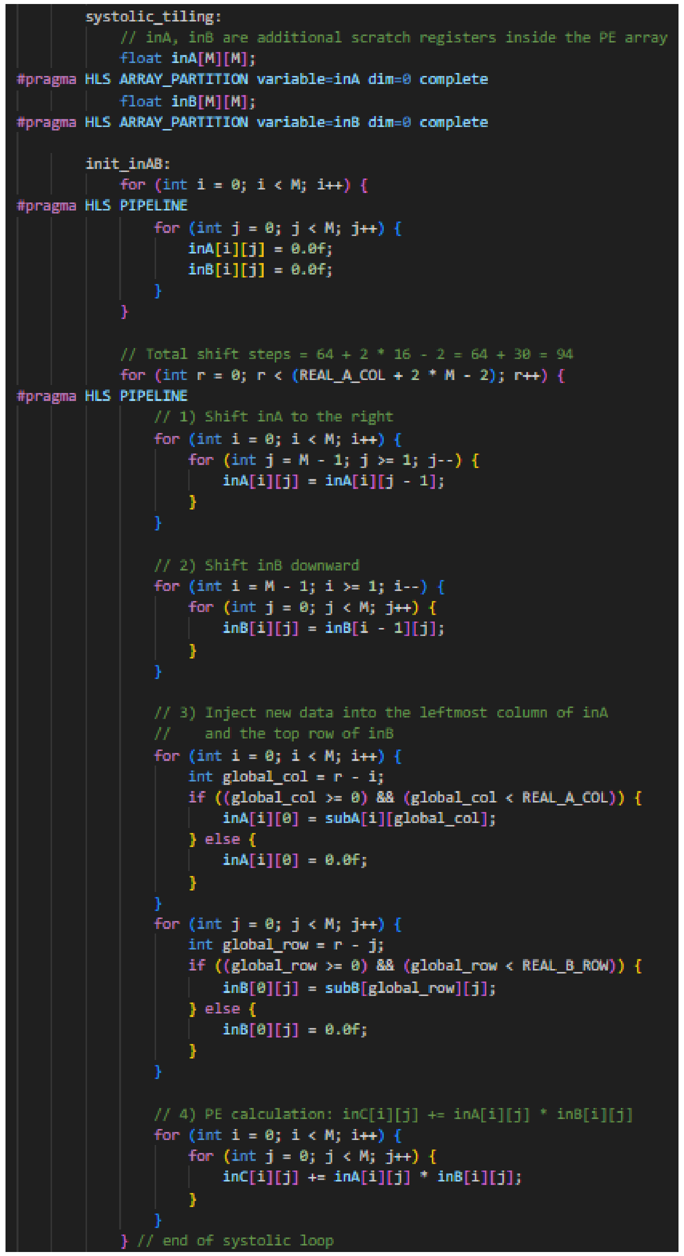
```
            //--------------------------------------------------------
            // (b) Retrieve the sub-block of data needed for this tile
            //     from localA, localB into subA, subB.
            //     Perform zero-padding if out of valid range.
            //--------------------------------------------------------
        read_subA:
            for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE
                for (int j = 0; j < REAL_A_COL; j++) { // j < 64
                    int globalRow = tileRow * M + i;  // 0..207
                    // real col = j
                    if (globalRow < REAL_A_ROW) {  // <197
                        subA[i][j] = A[globalRow][j];
                    } else {
                        subA[i][j] = 0.0f;
                    }
                }
            }

        read_subB:
            for (int i = 0; i < REAL_B_ROW; i++) {    // i < 64
#pragma HLS PIPELINE
                for (int j = 0; j < M; j++) {        // j < 16
                    int globalCol = tileCol * M + j; // 0..207
                    // real row = i
                    if (globalCol < REAL_B_COL) {   // <197
                        subB[i][j] = B[i][globalCol];
                    } else {
                        subB[i][j] = 0.0f;
                    }
                }
            }
```



```
            //--------------------------------------------------------
            // (d) Write the tile's computed result (inC) back to
            //     the corresponding location in C,
            //     only if it's within the valid 197×197 range.
            //--------------------------------------------------------
        store_tileC:
            for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE II=1
                for (int j = 0; j < M; j++) {
                    int globalRow = tileRow * M + i; // 0..207
                    int globalCol = tileCol * M + j; // 0..207
                    // Only write back if within the valid region of 197×197
                    if ((globalRow < REAL_A_ROW) && (globalCol < REAL_B_COL)) {
                        // Use += as in the original code
                        C[globalRow][globalCol] += inC[i][j];
                    }
                }
            }
```

Systolic Array Design with tiling & padding

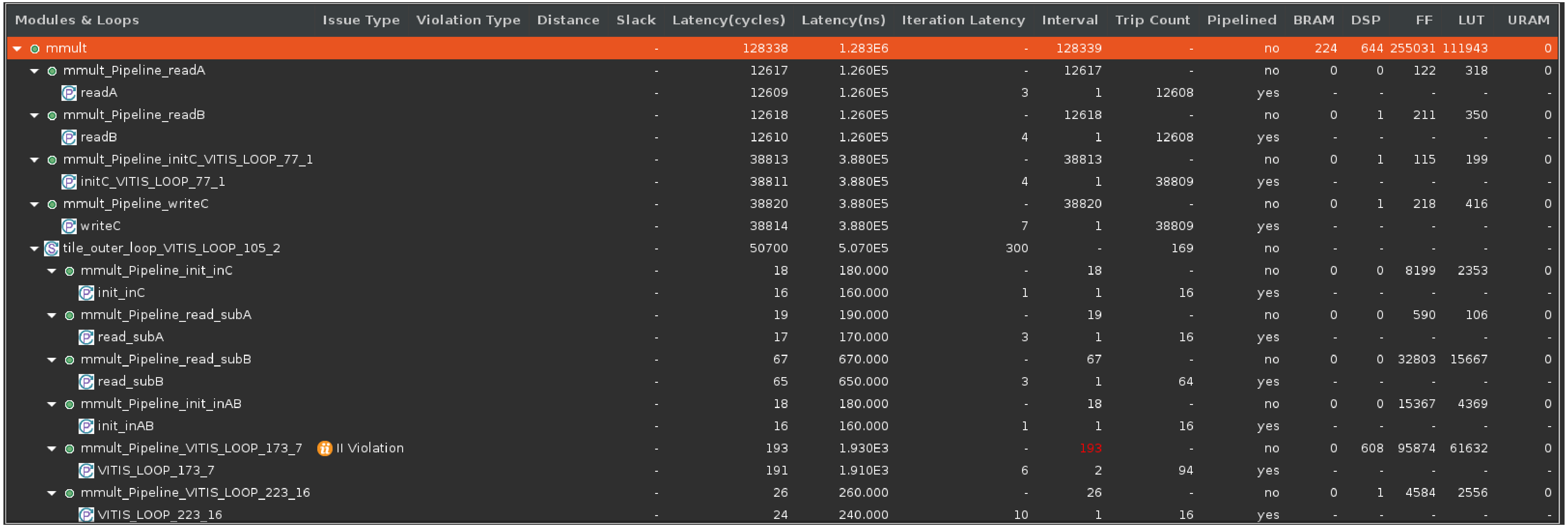




```
        systolic_tiling:
            // inA, inB are additional scratch registers inside the PE array
            float inA[M][M];
#pragma HLS ARRAY_PARTITION variable=inA dim=0 complete
            float inB[M][M];
#pragma HLS ARRAY_PARTITION variable=inB dim=0 complete

        init_inAB:
            for (int i = 0; i < M; i++) {
#pragma HLS PIPELINE
                for (int j = 0; j < M; j++) {
                    inA[i][j] = 0.0f;
                    inB[i][j] = 0.0f;
                }
            }

            // Total shift steps = 64 + 2 * 16 - 2 = 64 + 30 = 94
            for (int r = 0; r < (REAL_A_COL + 2 * M - 2); r++) {
#pragma HLS PIPELINE
                // 1) Shift inA to the right
                for (int i = 0; i < M; i++) {
                    for (int j = M - 1; j >= 1; j--) {
                        inA[i][j] = inA[i][j - 1];
                    }
                }

                // 2) Shift inB downward
                for (int i = M - 1; i >= 1; i--) {
                    for (int j = 0; j < M; j++) {
                        inB[i][j] = inB[i - 1][j];
                    }
                }

                // 3) Inject new data into the leftmost column of inA
                //    and the top row of inB
                for (int i = 0; i < M; i++) {
                    int global_col = r - i;
                    if ((global_col >= 0) && (global_col < REAL_A_COL)) {
                        inA[i][0] = subA[i][global_col];
                    } else {
                        inA[i][0] = 0.0f;
                    }
                }
                for (int j = 0; j < M; j++) {
                    int global_row = r - j;
                    if ((global_row >= 0) && (global_row < REAL_B_ROW)) {
                        inB[0][j] = subB[global_row][j];
                    } else {
                        inB[0][j] = 0.0f;
                    }
                }

                // 4) PE calculation: inC[i][j] += inA[i][j] * inB[i][j]
                for (int i = 0; i < M; i++) {
                    for (int j = 0; j < M; j++) {
                        inC[i][j] += inA[i][j] * inB[i][j];
                    }
                }
            } // end of systolic loop
```

Systolic Array Design with tiling & padding

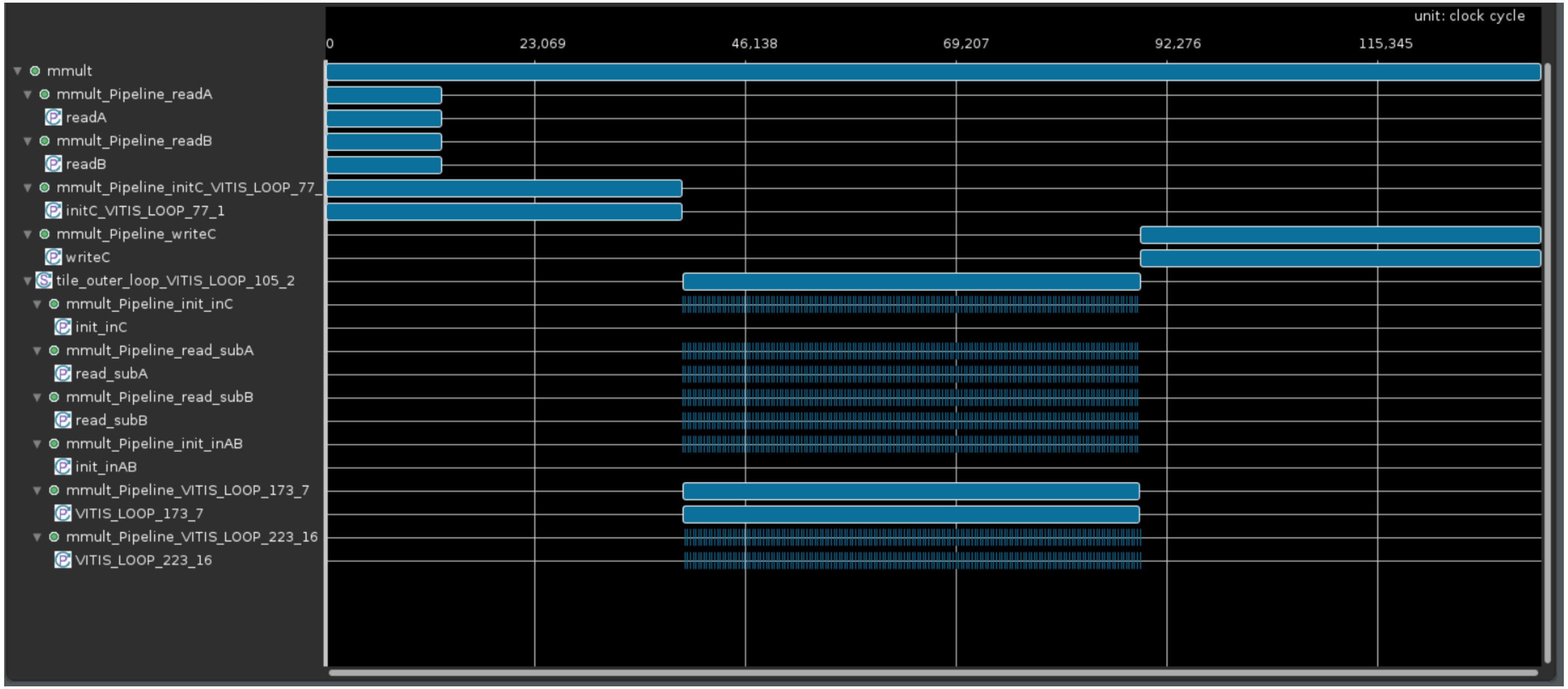
- Synthesis Report



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
mmult				-	128338	1.283E6	-	128339	-	no	224	644	255031	111943	0
mmult_Pipeline_readA				-	12617	1.260E5	-	12617	-	no	0	0	122	318	0
readA				-	12609	1.260E5	3	1	12608	yes	-	-	-	-	-
mmult_Pipeline_readB				-	12618	1.260E5	-	12618	-	no	0	1	211	350	0
readB				-	12610	1.260E5	4	1	12608	yes	-	-	-	-	-
mmult_Pipeline_initC_VITIS_LOOP_77_1				-	38813	3.880E5	-	38813	-	no	0	1	115	199	0
initC_VITIS_LOOP_77_1				-	38811	3.880E5	4	1	38809	yes	-	-	-	-	-
mmult_Pipeline_writeC				-	38820	3.880E5	-	38820	-	no	0	1	218	416	0
writeC				-	38814	3.880E5	7	1	38809	yes	-	-	-	-	-
tile_outer_loop_VITIS_LOOP_105_2				-	50700	5.070E5	300	-	169	no	-	-	-	-	-
mmult_Pipeline_init_inC				-	18	180.000	-	18	-	no	0	0	8199	2353	0
init_inC				-	16	160.000	1	1	16	yes	-	-	-	-	-
mmult_Pipeline_read_subA				-	19	190.000	-	19	-	no	0	0	590	106	0
read_subA				-	17	170.000	3	1	16	yes	-	-	-	-	-
mmult_Pipeline_read_subB				-	67	670.000	-	67	-	no	0	0	32803	15667	0
read_subB				-	65	650.000	3	1	64	yes	-	-	-	-	-
mmult_Pipeline_init_inAB				-	18	180.000	-	18	-	no	0	0	15367	4369	0
init_inAB				-	16	160.000	1	1	16	yes	-	-	-	-	-
mmult_Pipeline_VITIS_LOOP_173_7	II Violation			-	193	1.930E3	-	193	-	no	0	608	95874	61632	0
VITIS_LOOP_173_7				-	191	1.910E3	6	2	94	yes	-	-	-	-	-
mmult_Pipeline_VITIS_LOOP_223_16				-	26	260.000	-	26	-	no	0	1	4584	2556	0
VITIS_LOOP_223_16				-	24	240.000	10	1	16	yes	-	-	-	-	-

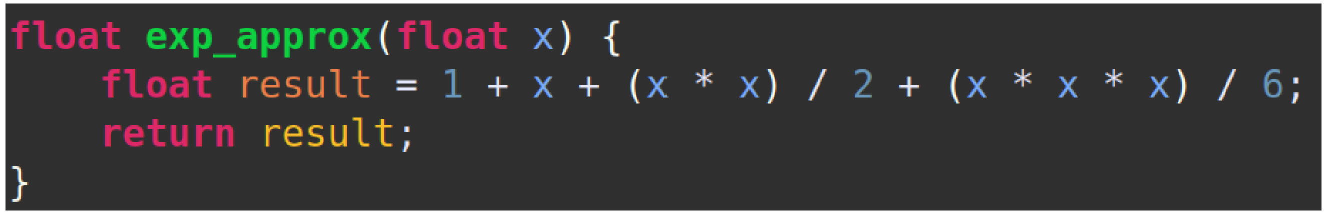
Systolic Array Design with tiling & padding

- C/RTL Co-simulation Timeline



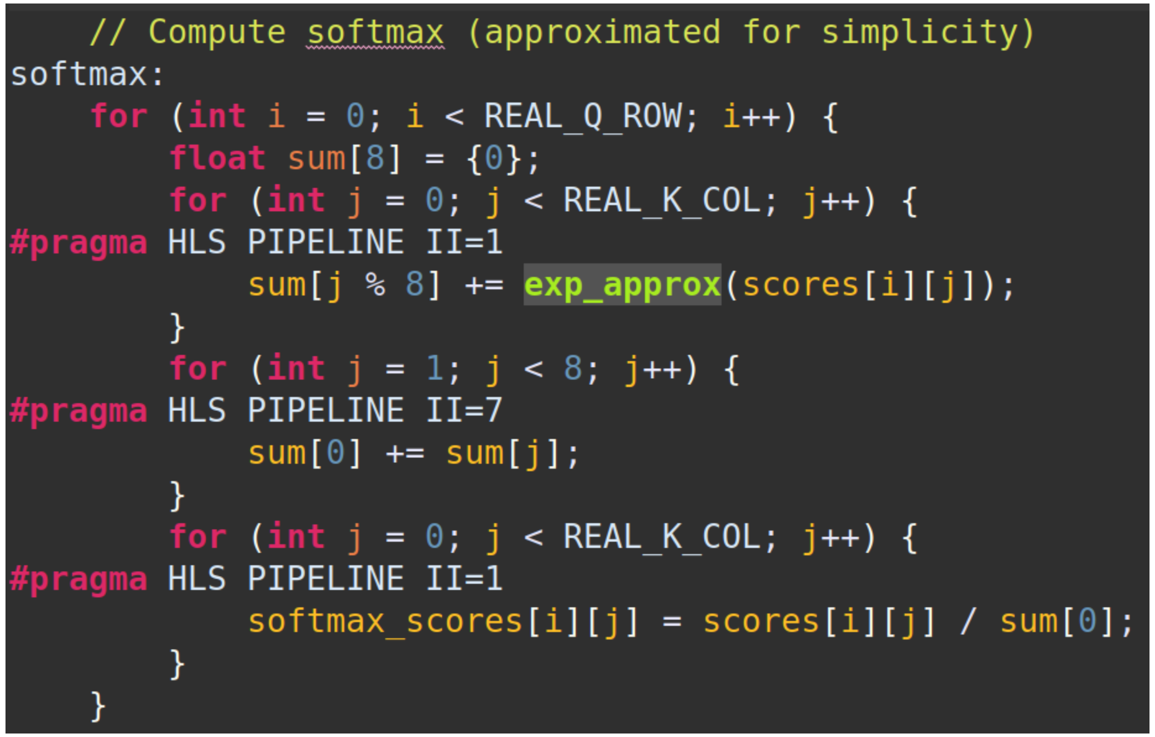
Softmax

- Exp(x)



```
float exp_approx(float x) {
    float result = 1 + x + (x * x) / 2 + (x * x * x) / 6;
    return result;
}
```

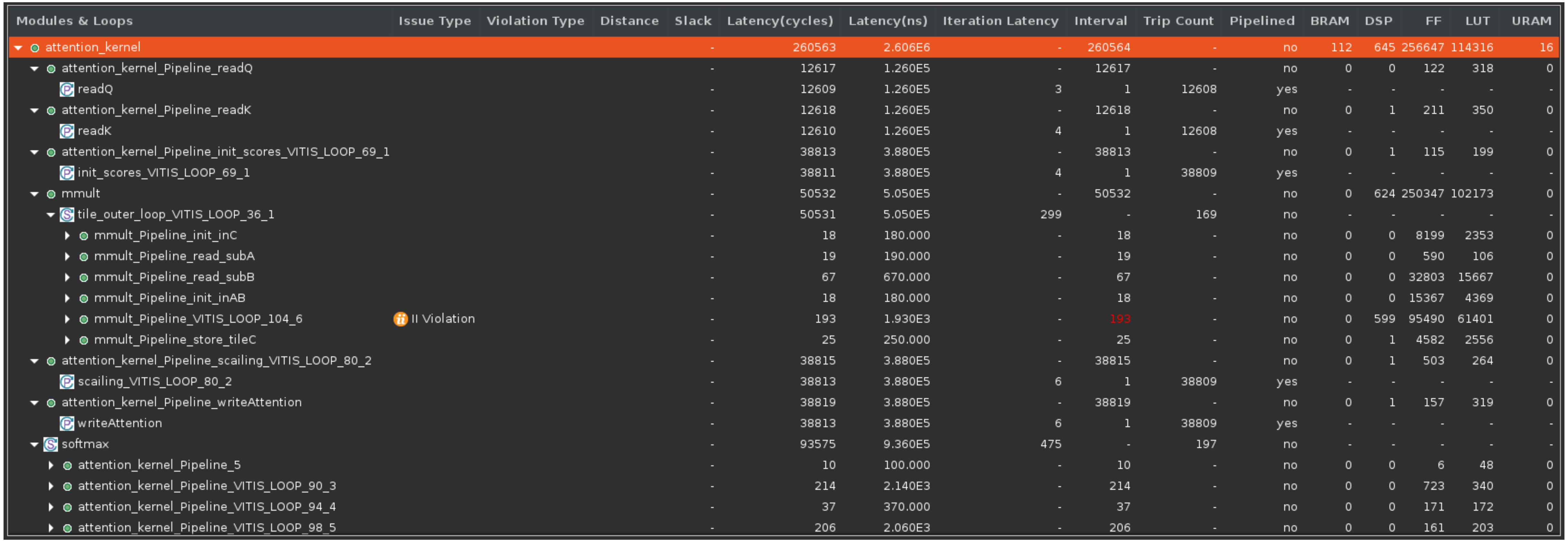
- Softmax()



```
    // Compute softmax (approximated for simplicity)
softmax:
    for (int i = 0; i < REAL_Q_ROW; i++) {
        float sum[8] = {0};
        for (int j = 0; j < REAL_K_COL; j++) {
#pragma HLS PIPELINE II=1
            sum[j % 8] += exp_approx(scores[i][j]);
        }
        for (int j = 1; j < 8; j++) {
#pragma HLS PIPELINE II=7
            sum[0] += sum[j];
        }
        for (int j = 0; j < REAL_K_COL; j++) {
#pragma HLS PIPELINE II=1
            softmax_scores[i][j] = scores[i][j] / sum[0];
        }
    }
```

Attention Kernel

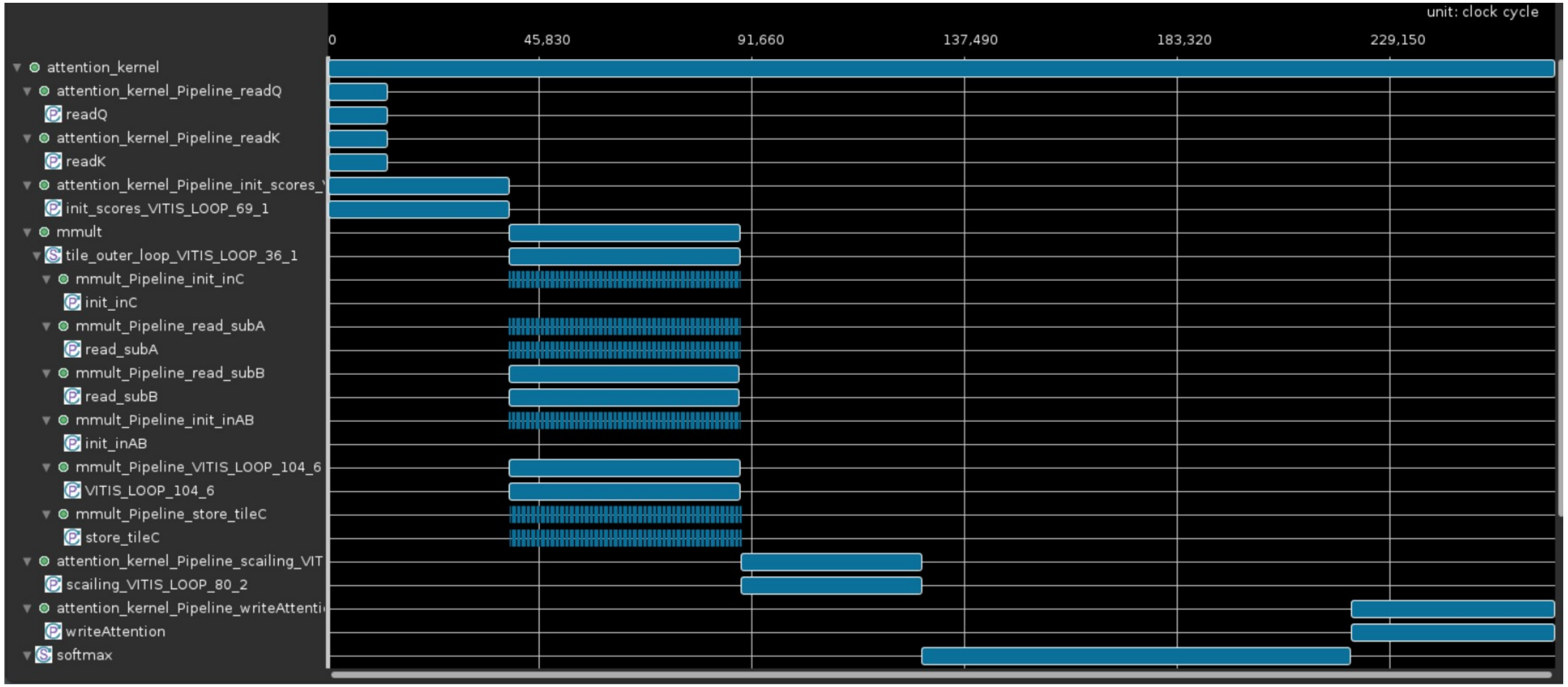
- Synthesis Report



Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
attention_kernel				-	260563	2.606E6	-	260564	-	no	112	645	256647	114316	16
attention_kernel_Pipeline_readQ				-	12617	1.260E5	-	12617	-	no	0	0	122	318	0
readQ				-	12609	1.260E5	3	1	12608	yes	-	-	-	-	-
attention_kernel_Pipeline_readK				-	12618	1.260E5	-	12618	-	no	0	1	211	350	0
readK				-	12610	1.260E5	4	1	12608	yes	-	-	-	-	-
attention_kernel_Pipeline_init_scores_VITIS_LOOP_69_1				-	38813	3.880E5	-	38813	-	no	0	1	115	199	0
init_scores_VITIS_LOOP_69_1				-	38811	3.880E5	4	1	38809	yes	-	-	-	-	-
mmult				-	50532	5.050E5	-	50532	-	no	0	624	250347	102173	0
tile_outer_loop_VITIS_LOOP_36_1				-	50531	5.050E5	299	-	169	no	-	-	-	-	-
mmult_Pipeline_init_inC				-	18	180.000	-	18	-	no	0	0	8199	2353	0
mmult_Pipeline_read_subA				-	19	190.000	-	19	-	no	0	0	590	106	0
mmult_Pipeline_read_subB				-	67	670.000	-	67	-	no	0	0	32803	15667	0
mmult_Pipeline_init_inAB				-	18	180.000	-	18	-	no	0	0	15367	4369	0
mmult_Pipeline_VITIS_LOOP_104_6	II Violation			-	193	1.930E3	-	193	-	no	0	599	95490	61401	0
mmult_Pipeline_store_tileC				-	25	250.000	-	25	-	no	0	1	4582	2556	0
attention_kernel_Pipeline_scaling_VITIS_LOOP_80_2				-	38815	3.880E5	-	38815	-	no	0	1	503	264	0
scaling_VITIS_LOOP_80_2				-	38813	3.880E5	6	1	38809	yes	-	-	-	-	-
attention_kernel_Pipeline_writeAttention				-	38819	3.880E5	-	38819	-	no	0	1	157	319	0
writeAttention				-	38813	3.880E5	6	1	38809	yes	-	-	-	-	-
softmax				-	93575	9.360E5	475	-	197	no	-	-	-	-	-
attention_kernel_Pipeline_5				-	10	100.000	-	10	-	no	0	0	6	48	0
attention_kernel_Pipeline_VITIS_LOOP_90_3				-	214	2.140E3	-	214	-	no	0	0	723	340	0
attention_kernel_Pipeline_VITIS_LOOP_94_4				-	37	370.000	-	37	-	no	0	0	171	172	0
attention_kernel_Pipeline_VITIS_LOOP_98_5				-	206	2.060E3	-	206	-	no	0	0	161	203	0

Attention Kernel

- C/RTL Co-simulation Timeline



Individual Contribution

- 黃聖崴
  - Systolic array
  - Application integration
- 陳灝
  - Host code prepare
  - OpenCL, wrapper
- 祝華劭
  - Softmax
  - Application integration