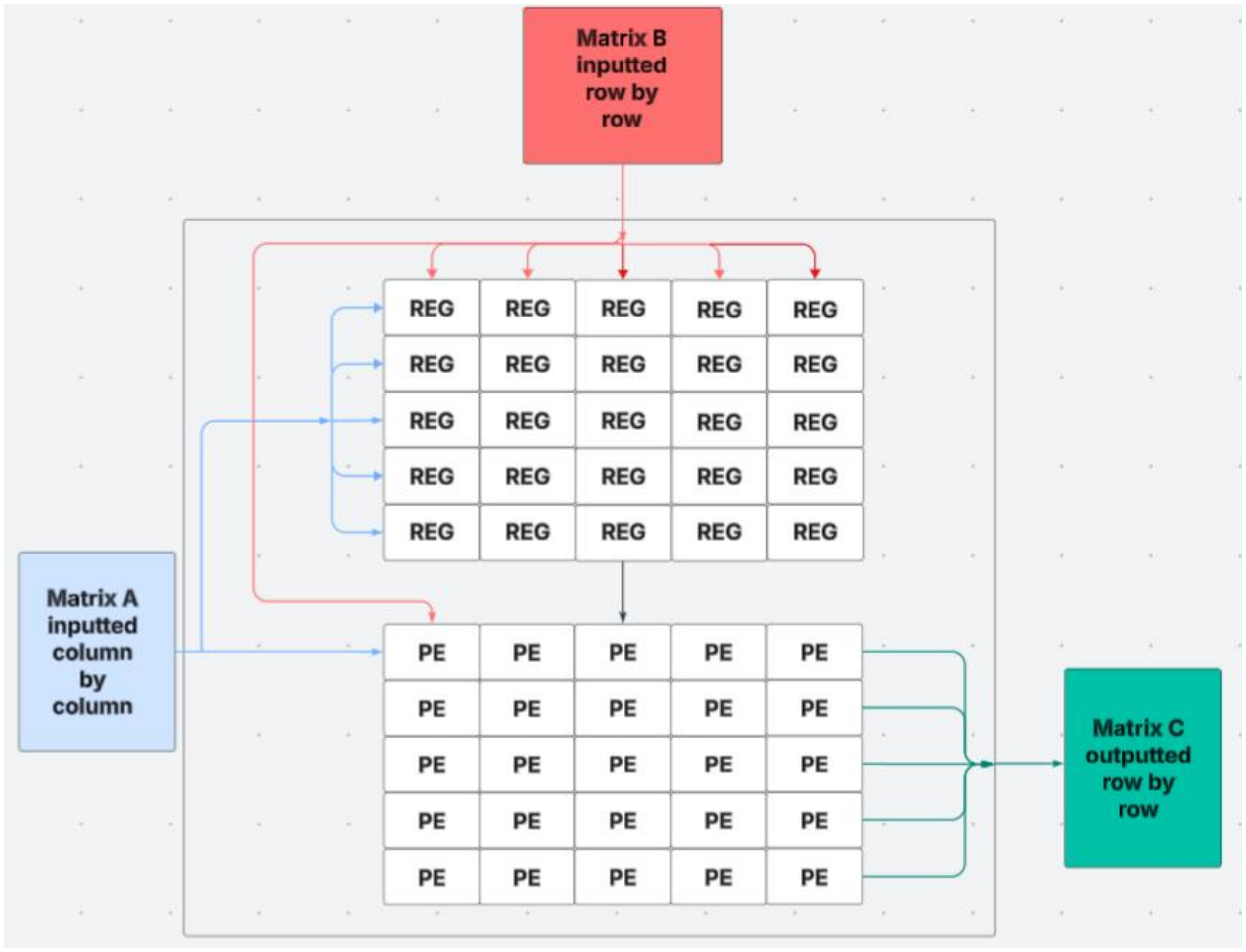
Abdelrahman Osama
01091960743
27/7/2025

STMicroelectronics LAB 0
Systolic Array for Applying Matrix Multiplication

1. Architecture


Matrix B inputted row by row
REG
PE
Matrix A inputted column by column
Matrix C outputted row by row

2. <u>Explanation of operation</u>

I have created 2D register arrays (A_reg and B_reg) to enable pipelined dataflow through the systolic array structure. These registers buffer the input matrices A and B, allowing elements to propagate vertically and horizontally across the grid of processing elements. Each processing element accumulates the product of corresponding values from A_reg and B_reg into a partial_sum array, which holds the intermediate and final results of the matrix multiplication. A special case is the top-left processing element (PE[0][0]), which does not need to wait for registered values — it directly computes its output from the current input ports, reducing latency by one cycle for that position. The design is fully parameterized by N_SIZE, allowing it to scale to any square matrix dimension. Control signals such as counter, reg_idx, and valid_out coordinate the loading of inputs, accumulation of results, and streaming of the output matrix C, ensuring correct timing and synchronization throughout the operation.

❖ The operation can be broken into three phases for extra details:

1. Data Load
   - Matrix **A** and **B** are loaded into internal shift registers (A_reg and B_reg).
   - Each cycle, one column of A and one row of B are streamed in.
   - This continues until the full matrices are buffered.

2. Multiply-Accumulate Phase
   - The systolic MAC network begins accumulating partial products.
   - reg_idx controls which register slice of A_reg and B_reg is used.
   - All partial_sum[i][j] values are updated over several cycles as data propagates diagonally across the array.
   - The top-left cell partial_sum[0][0] is handled separately using direct inputs

3. Output Phase
   - The valid_out signal is asserted.
   - Each clock cycle, a row from the resulting matrix C is output from partial_sum.

From my testing, I have concluded the following timing scheme:

Phase	Cycle Range
Data Load	0 to N_SIZE-1
MAC Accumulation	N_SIZE to N_SIZE+2
Output Start	Cycle = (2 × N_SIZE) – 1
Output End	Cycle = (3 × N_SIZE) – 2

Additional Notes:

I needed to synchronize the reg_idx signal in a specific way so, I created a valid_in_d signal which is just a delayed valid_in signal which then I use both to produce a valid_in_rise signal to give me the rising edge of the valid_in.

For my testbench, I have included three test cases : 3x3, 4x4, and 5x5. The 5x5 case is enabled by default. To run a different test, change N_SIZE then comment out the 5×5 configuration and uncomment the desired one. I have also added self-checking functionality where I compare the output of my DUT at the rise of the valid_out signal with the expected result of the multiplication of the two input matrices. If an error occurs, an integer called error_count is incremented and a display message is shown on the terminal indicating the location of such error.

3. Simulation

I ran all my simulations on QuestaSim 2023.3. In my sim directory, I have included two additional files: run.do and wave.do

The wave.do is just showing the internal signals which I used when debugging. It's mainly for the 5x5 test case.

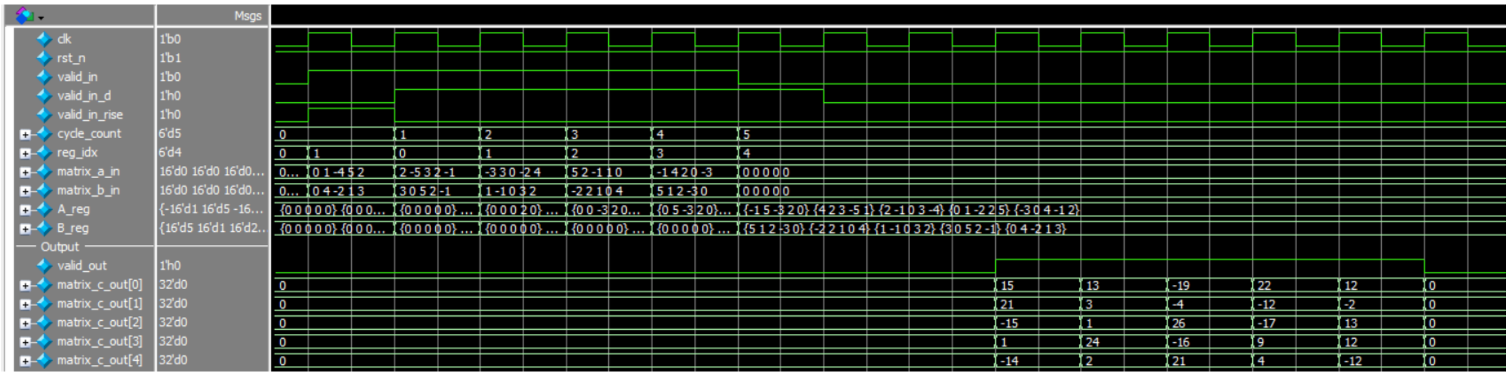
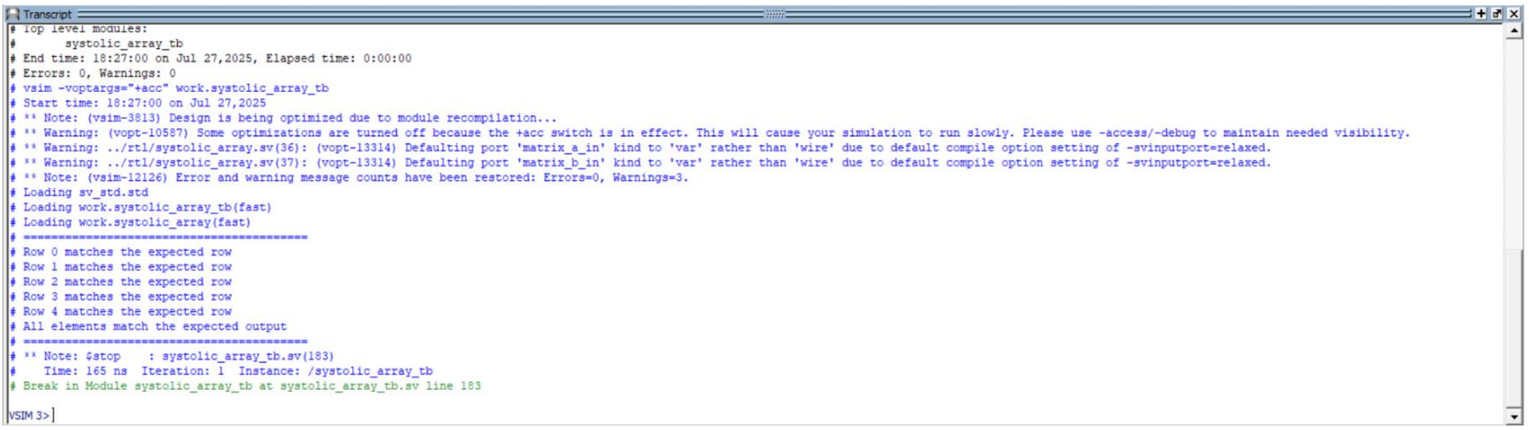


Fig. 1. QuestaSim Waveform

Fig. 1 shows the inputting of both matrices across five clock cycles along with the valid_in signal. It also shows the outputting of matrix_c_out with the valid_out signal. The first row of the output matrix appears on the bus three clock cycles after the last A column and B row were inputted.



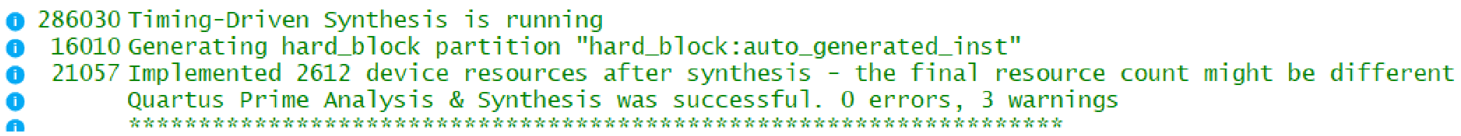
```
# Top level modules:
#       systolic_array_tb
# End time: 18:27:00 on Jul 27,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.systolic_array_tb
# Start time: 18:27:00 on Jul 27,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# ** Warning: (vopt-10587) Some optimizations are turned off because the +acc switch is in effect. This will cause your simulation to run slowly. Please use -access/-debug to maintain needed visibility.
# ** Warning: ../rtl/systolic_array.sv(36): (vopt-13314) Defaulting port 'matrix_a_in' kind to 'var' rather than 'wire' due to default compile option setting of -svinputport=relaxed.
# ** Warning: ../rtl/systolic_array.sv(37): (vopt-13314) Defaulting port 'matrix_b_in' kind to 'var' rather than 'wire' due to default compile option setting of -svinputport=relaxed.
# ** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=3.
# Loading sv_std.std
# Loading work.systolic_array_tb(fast)
# Loading work.systolic_array(fast)
# ========================================
# Row 0 matches the expected row
# Row 1 matches the expected row
# Row 2 matches the expected row
# Row 3 matches the expected row
# Row 4 matches the expected row
# All elements match the expected output
# ========================================
# ** Note: $stop    : systolic_array_tb.sv(183)
#    Time: 165 ns  Iteration: 1  Instance: /systolic_array_tb
# Break in Module systolic_array_tb at systolic_array_tb.sv line 183

VSIM 3>
```

Fig. 2. QuestaSim Terminal

Fig. 2 shows the output on the terminal for the 5x5 test case. Every element matched the expected output.

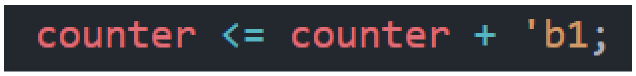
4. Synthesis



```
286030 Timing-Driven Synthesis is running
 16010 Generating hard_block partition "hard_block:auto_generated_inst"
 21057 Implemented 2612 device resources after synthesis - the final resource count might be different
       Quartus Prime Analysis & Synthesis was successful. 0 errors, 3 warnings
       ******************************************************************
```

Fig. 3. Quartus Prime Report

I have used Quartus Prime 20.1 to synthesize and compile my design. Fig. 3 shows the successful compilation of the design with no errors. The warnings were just the truncation of the “**b1**” I use to increment my counters from 32 bits to match my counters’ width.



```
counter <= counter + 'b1;
```

Fig. 4. Code Snippet

5. Final Notes

I have complied with the design requirements of having all functionality in one file “systolic_array.sv” as well as the provided port list, and parameters.

The design provided correct outputs for every test case I tried including negative numbers and zeros in the input matrices.