Assignment 2: Matrix Multiplication

In this assignment we're building a matrix multiplication algorithm in RTL SystemVerilog. You will be analyzing the performance of the implementation in terms of cycles, timing, and resources. Begin by compiling the provided C code to generate data from matrices A, B, and C.

```
void matmul( int *A[], int *B[], int *C[], int N ) {
    for ( int i=0; i<N; i++ ) {
       for ( int j=0; j<N; j++ ) {
            C[i][j] = 0;
            for ( int k=0; k<N; k++ ) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

Goals:

- Parameterize the matrix size NxN using parameters in your module declaration
- Read matrices A and B from Block RAMS, and write matrix C to a Block RAM
- Report on performance results and improvements

Implementation:

- Run the provided C program of the matmul function to generate random data for two 8x8 matrices, A & B, and the matrix multiplication results for matrix C. The output txt files will each contain 64 values in 32-bit hex format.
- Implement the matmul module in SystemVerilog using Block RAMs to store the data. The Block RAMs should be instantiated in a matmul_top wrapper module. The implementation should be relatively similar to the Vectorsum example from class.
- The matmul module should be implemented using the 2-process method (always_ff and always_comb), and should contain a finite-state machine (FSM).
- Optimize the FSM so that on average the main loop iteration is only 1-cycle (e.g. one state), so that the number of total cycles is ~N^3

Verification:

- Create a testbench that reads the matrices A & B from text files and write them to input Block RAMS. Likewise, it should read the C matrix from the

output Block RAM and compare results against the known data from your C program.

- Configure your testbench with the following generic parameters:
    - 32-bit wide data
    - 64-element Block RAM
    - Frequency of 100MHZ (10ns clock period)
- Create a simulation.do file that compiles the SystemVerilog files, sets up the wave form, and runs the simulation. You should be able to run "vsim -do matmul_sim.do" from the commandline.
- Run the simulation in modelsim and verify the design is bit-true accurate.

Synthesis:

- Compile the design in Synplify Premier to get high-level resource results and timing information.
- Target the Intel Cyclone IV-E FPGA
- Get the resource utilization and timing information (max frequency)

Reporting:

- Submit a PDF file with simulation and synthesis results with the following:
    - Simulation results:
        - Clock cycle count
        - Errors reported (if any)
    - Synthesis results (some might not apply):
        - Maximum frequency
        - Registers / LUTs / Logic Elements
        - Memory utilization
        - Multipliers (DSPs)
        - Worst path (timing analysis)
        - Schematic architecture (RTL)

Turn in your designs with the report. Your file should be zipped, and should include the SystemVerilog files, synthesis, simulation, and input/output files. **PLEASE MAKE SURE TO REMOVE ALL THE INTERMEDIATE WORK DIRECTORIES.**