Homework Assignment 4
Hardware Accelerator for Matrix Multiplication using Vivado HLS

Introduction:

Using Vivado HLS, design a hardware accelerator, called MATRIX_MUL, that calculates a product of two matrices, based on the following description of this operation from Wikipedia.

For two matrices, the \( n \times m \) matrix \( A \), and the \( m \times p \) matrix \( B \):

\[
A = \begin{pmatrix}
A_{11} & A_{12} & \cdots & A_{1m} \\
A_{21} & A_{22} & \cdots & A_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
A_{n1} & A_{n2} & \cdots & A_{nm}
\end{pmatrix}, \quad B = \begin{pmatrix}
B_{11} & B_{12} & \cdots & B_{1p} \\
B_{21} & B_{22} & \cdots & B_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
B_{m1} & B_{m2} & \cdots & B_{mp}
\end{pmatrix},
\]

where necessarily the number of columns in \( A \) is equal to the number of rows in \( B \) (in this case \( m \)), the matrix product, denoted by \( AB \), is the \( n \times p \) matrix:

\[
AB = \begin{pmatrix}
(AB)_{11} & (AB)_{12} & \cdots & (AB)_{1p} \\
(AB)_{21} & (AB)_{22} & \cdots & (AB)_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
(AB)_{n1} & (AB)_{n2} & \cdots & (AB)_{np}
\end{pmatrix}
\]

where \( AB \) has entries defined as follows:

\[
(AB)_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}
\]

Assumptions:
Each component of the matrices \( A \) and \( B \), e.g., \( A_{11}, B_{22} \), is an 8-bit unsigned integer. Each component of the matrix \( AB \), e.g., \( AB_{11}, AB_{22} \), is a \((16+lm)\)-bit unsigned integer.

\[
m = 2^{lm}, \text{ where } lm \text{ is an integer, such that } 1 \leq lm \leq mln, \text{ e.g., } m = 2^3 = 8
\]

\[
n = 2^{ln}, \text{ where } ln \text{ is an integer, such that } 1 \leq ln \leq mln, \text{ e.g., } n = 2^4 = 16
\]

\[
p = 2^{lp}, \text{ where } lp \text{ is an integer, such that } 1 \leq lp \leq mlp, \text{ e.g., } p = 2^5 = 32.
\]

Components of \( A \) and \( B \) are initially stored in \textbf{RAMs}, row by row, i.e., in the order \( A_{11}, A_{12}, \ldots, A_{1m}, A_{21}, A_{22}, \ldots, A_{2m}, \ldots, A_{n1}, A_{n2}, \ldots, A_{nm} \) and \( B_{11}, B_{12}, \ldots, B_{1p}, B_{21}, B_{22}, \ldots, B_{2p}, \ldots, B_{m1}, B_{m2}, \ldots, B_{mp} \) respectively.

Components of \( AB \) should be stored in \textbf{RAM}, row by row, i.e., in the order \( AB_{11}, AB_{12}, \ldots, AB_{1p}, AB_{21}, AB_{22}, \ldots, AB_{2p}, \ldots, AB_{n1}, AB_{n2}, \ldots, AB_{np} \).
Rows of matrices can be separated by unused memory locations, if that simplifies your design or testing.

The circuit should be optimized for the minimum execution time. In case of circuits with the comparable execution time, the preference should be given to the circuit with the smaller area.

mlm, mln, and mlp should be treated as constants and set to 7.

Values of parameters lm, ln, and lp should be passed to the HLS IP core using AXI-Lite interface.

RAMs storing Matrices A and B should be initialized at the start of computations, using AXI-Stream interface.

RAM storing matrix AB should be read at the end of computations using AXI-Stream Interface.

Your circuit should follow the high-level block diagram given below:

*Memory = ACP, OCM, or DRAM
Tasks:

Task 1

Write the C code for the MATRIX_MUL in Vivado HLS environment.

Please separate
- initializing matrices A and B,
- performing calculations,
- printing results.
Each of the above operations should be a separate function called from main().

Define \( l_m \), \( l_n \), \( l_p \) as constants using \#define.
Define \( m \), \( n \), \( p \) as constants, dependent on \( l_m \), \( l_n \), and \( l_p \), using \#define. Calculate power of 2 using shift left.

Do not read any of these values using scanf().

Initialize arrays A and B with pseudorandom values in the range between 0 and 255.

Make the arrays A and B of the type u8 or unsigned char.
Make the array AB of the type u32 or unsigned int.

Task 2

Write the test bench in C/ C++ and add it to Vivado HLS. The test bench should be able to calculate the product of two given matrixes using both software and the HLS hardware solution, comparing the two to ensure successful operation.

Task 3

Run C Simulation and make sure that “Test passed” message is displayed in the console.

Task 4

Run the C Synthesis with the default settings and obtain the information regarding timing, latency and resource utilization estimates.

Task 5

Run C/ RTL Cosimulation and make sure that you get the “Pass” status for the VHDL in the Cosimulation report.

Task 6

Apply various directives to improve the execution time of the design.
Task 8

Apply directives in Vivado HLS design, which infer the AXI-Stream interface for the A, B and AB matrixes, and AXI-Lite interface for lm, ln, and lp.

Task 9

Export Vivado HLS as an IP core compatible with the Vivado IP Catalog and IP Integrator.

Task 10

Create a block design and include all IPs into the design including Vivado HLS IP core as shown in the provided block diagram.

Task 11

Debug and optimize your main C file.

Verify results returned by your software/hardware implementations, by comparing them with results generated by your purely software implementation, for several different values of parameters lm, ln, and lp.

Task 12

Measure and tabulate the total execution time of matrix multiplication for
- Purely software implementation
- All developed software/hardware implementations.

Modify values of parameters lm, ln, lp, by changing constants in the main C program only.

Repeat measurements for the following values of the parameters: 

\[ lm = mlm = 7, \ ln = lp = 1..mln. \]

Task 13

Calculate and tabulate the speed-up of your most efficient hardware/software implementation over the purely software implementation as a function of \( lnp=ln=lp \), for \( lm = mlm \).