Design a hardware accelerator, called MATRIX_MUL, calculating 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:

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

**Make the following assumptions:**

Each component of the matrices $A$ and $B$, e.g., $A_{11}$, $B_{22}$, is an 8-bit unsigned integer.

$m = 2^{lm}$, where $lm$ is an integer, such that $1 \leq lm \leq mlm$, e.g., $m = 2^3 = 8$

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

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

Components of $A$ and $B$ are initially stored in 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.

**Individual students should use one single-port distributed RAM for $A$ and one single-port distributed RAM for $B$.**

**Teams of two-students should use one dual-port BRAM for $A$ and one-dual port BRAM for $B$.**

Components of $AB$ should be stored in RAM, row by row, i.e., in the order $AB_{11}, AB_{12}, \ldots, AB_{1p}, AB_{21}, AB_{22}, \ldots, AB_{2p}, \ldots, AB_{n1}, AB_{n2}, \ldots, AB_{np}$.
Each component of the matrix \( \mathbf{AB} \), e.g., \( AB_{11}, AB_{22} \), is a \((16 + lm)\)-bit unsigned integer. **Individual students should use a single-port distributed RAM.**

Teams of two-students should use a single-port BRAM.

**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.

The interface of the circuit is defined using the following table of input/output ports:

<table>
<thead>
<tr>
<th>Port</th>
<th>Mode</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>Input</td>
<td>1</td>
<td>System clock.</td>
</tr>
<tr>
<td>resetn</td>
<td>Input</td>
<td>1</td>
<td>Asynchronous system reset active low.</td>
</tr>
<tr>
<td>s</td>
<td>Input</td>
<td>1</td>
<td>Operating mode: 0 = waiting for data / reading results, 1 = processing.</td>
</tr>
<tr>
<td>A_addr</td>
<td>Input</td>
<td>( m_{lm} + m_{ln} )</td>
<td>Address determining a location in the RAM used to store matrix A.</td>
</tr>
<tr>
<td>A_wr</td>
<td>Input</td>
<td>1</td>
<td>Write signal for the RAM used to store matrix A.</td>
</tr>
<tr>
<td>B_addr</td>
<td>Input</td>
<td>( m_{lp} + m_{lm} )</td>
<td>Address determining a location in the RAM used to store matrix B.</td>
</tr>
<tr>
<td>B_wr</td>
<td>Input</td>
<td>1</td>
<td>Write signal for the RAM used to store matrix B.</td>
</tr>
<tr>
<td>done</td>
<td>Output</td>
<td>1</td>
<td>Control signal indicating that the output is ready.</td>
</tr>
<tr>
<td>AB_out</td>
<td>Output</td>
<td>( 16 + m_{lm} )</td>
<td>((16 + lm))-bit component of the matrix AB.</td>
</tr>
<tr>
<td>AB_addr</td>
<td>Input</td>
<td>( m_{lp} + m_{ln} )</td>
<td>Address determining a location in the RAM used to store matrix AB.</td>
</tr>
</tbody>
</table>

\( lm \) \( ln \) \( lp \)

\( lm \) \( ln \) \( lp \)

\( \text{The actual value of } lm = \log_2 m, \text{ in the range } [0,7]. \)

\( \text{The actual value of } ln = \log_2 n, \text{ in the range } [0,7]. \)

\( \text{The actual value of } lp = \log_2 p, \text{ in the range } [0,7]. \)

RAMs storing Matrices \( \mathbf{A} \) and \( \mathbf{B} \) should be initialized at the start of computations, using ports \( A_{\text{in}}, A_{\text{addr}}, A_{\text{wr}}, B_{\text{in}}, B_{\text{addr}}, \) and \( B_{\text{wr}} \).

RAM storing matrix \( \mathbf{AB} \) should be read at the end of computations using ports \( AB_{\text{out}}, AB_{\text{addr}} \).

**Complete the following tasks:**

**Task 1**

Draw a block diagram of the Datapath of the MATRIX_MUL circuit. Clearly specify

- names, widths and directions of all buses
- names, widths and directions of all inputs and outputs of the logic components.

Please note that you can use connections by name.
Task 2

Draw an interface of the MATRIX_MUL circuit with the division into the Datapath and Controller.

Task 3

Draw an ASM chart describing the Controller of this circuit.

Task 4

Derive a formula for the execution time of your circuit (expressed in clock cycles), as a function of the parameters \( m \), \( n \), and \( p \).

Please make sure that this formula matches:
- your choice of the kind of memories (single-port vs. dual port), used to store matrices A, B, and AB
- your design for the datapath
- your design for the controller.

Task 5

Write the reference C code for the MATRIX_MUL.

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 \( lm \), \( ln \), \( lp \) as constants using #define.
Define \( m \), \( n \), \( p \) as constants, dependent on \( lm \), \( ln \), and \( lp \), 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 6

Write the VHDL code of the top-level component that includes the datapath and controller. Use generics for the parameters \( mlm \), \( mln \) and \( mlp \).
Task 7

Write the testbench of your design. Your testbench should be able to verify the top-level design for the following values of inputs denoting matrix dimensions: \( lm = 3, \ ln = 2, \ lp = 3 \).

Task 8

Synthesize and implement your design for the following values of generics:

- \( mlm = mln = mlp = 6 \) for individual students
- \( mlm = mln = mlp = 7 \) for teams of two students

Report the results for resource utilization, minimum clock, maximum frequency, and power consumption after implementation.

*Note: In case your synthesis and implementation time exceeds one hour, please feel free to use lower values of these parameters.*

Task 9

- Perform the timing analysis.
- Find out the critical path in your design and adjust your clock constraints accordingly.
- Run the timing simulation with maximum possible frequency and check the functional correctness of your design.
- Mark the most likely critical path in your block diagram.

Task 10

Select and implement the most efficient way of communication between your hardware accelerator and selected memory of Processing System (DRAM or OCM).

For individual students: implementing one method of communication is sufficient.

For teams of two students: please implement two substantially different methods of communication, e.g.,

- using DMA vs. Central DMA vs. no DMA,
- using HP interface vs. Master GP interface vs. Slave GP interface vs. ACP interface,
- using simple vs. scatter-gather DMA transfer.

Any difference mentioned in any line above is sufficient.

Please make sure to support passing the parameters \( lm, \ ln, \) and \( lp \), using one of the following options, independent of the major communication method used for the transfer of the input and output matrices:

**Option 1:** Parameters \( lm, \ ln, \) and \( lp \) are passed using AXI_Lite.

**Option 2:** Parameters \( lm, \ ln, \) and \( lp \) are passed in the header of input data.
Task 11

Create an IP of your functionally verified IP, and extend it with the VHDL code necessary to support the required interface that matches your communication method.

Task 12

Test the operation of your extended IP using a comprehensive testbench.

Task 13

Develop and test the entire system, including its hardware and software parts.

Do your best to choose the optimal values of parameters, such as burst size and packet size.

Please make sure to include an AXI timer to measure the execution time of your matrix multiplication.

Task 14

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 15

Measure and tabulate the total execution time of matrix multiplication for

a. purely software implementation
b. 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:

$l m = mlm$, $ln = lp = 1..mln$.

Task 16

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 $lm = mlm$. 
Deliverables:

Please create a top-level folder containing the following subfolders with deliverables for the respective tasks:

1_block_diagram (Task 1)
2_interface (Task 2)
3_ASM (Task 3)
4_timing_formulas (Task 4)
5_C (Task 5)
6_VHDL (Task 6)
7_verification (including testbench and results of verification) (Task 7)
8_results (including resource utilization, minimum clock period, maximum clock frequency, and power consumption after implementation) (Task 8)
9_timing_analysis (Task 9)
10_hardware_accelerator (Task 11) (VHDL code of your hardware accelerator and its testbench; the results of the verification)
10_block_design (your final block design obtained after performing Tasks 10-13)
12_main_program_verification (your main program, comparing outputs generated using purely software implementation with outputs generated using software/hardware implementation(s); the results of the verification) (Task 14)
13_timing_measurements (the summary of your timing measurements using AXI Timers) (Task 15)
14_speed_up_parameters (the summary of the obtained speed-up and its dependence on the matrix dimensions, and other parameters of your implementation, such as burst size) (Task 16).

Place files developed as a result of particular tasks in the corresponding folders. Please include only files necessary to reproduce the results. Please make sure to provide short summaries of major results and observations in the form of PDF, text, or MS Word files.

Please zip your top-level folder. Please submit this zip file only.