Systolic Array implementation in Verilog

Background – Matrix Multiplication

Matrix multiplication is a major part of the computation for many applications. Here we consider a 3x3 matrix multiplication: C = AxB + D

For simplicity, for this lab we will ignore D and consider only C = AxB only. Eg:

```
for (int k = 0; k < 3; k++) {
  for (int j = 0; j < 3; j++) {
    for (int i = 0; i < 3; i++) {
        C[i,j] += A[i,k] * B[k,j];
        }
  }
}
```

Background – Processing Elements

A Processing Element (PE) is a basic ALU unit that we will use in this systolic array implementations.

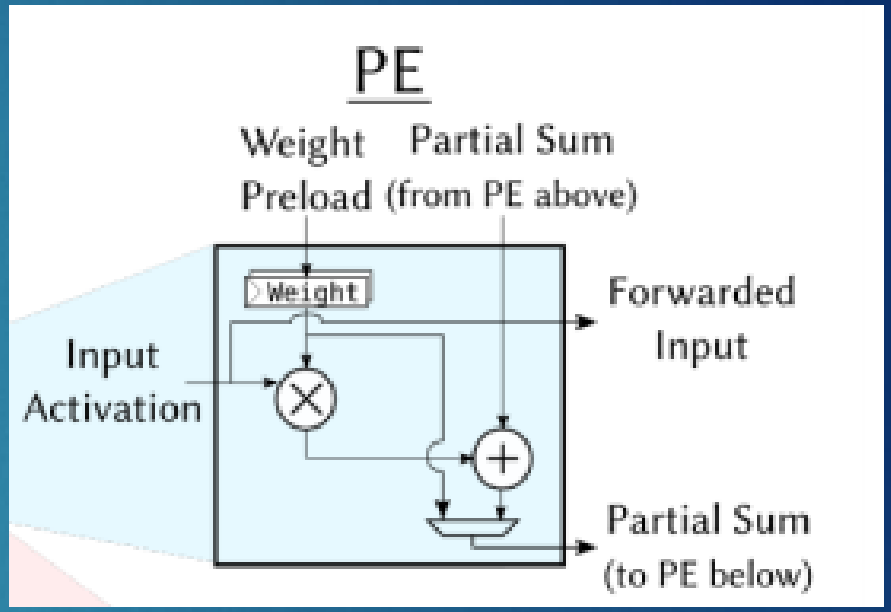
A basic PE has 3 data inputs, 3 data outputs.
Data Inputs:

- Input Activation (a)
- Weight Preload (b)
- Partial Sum from PE above (cin)

Data Outputs:

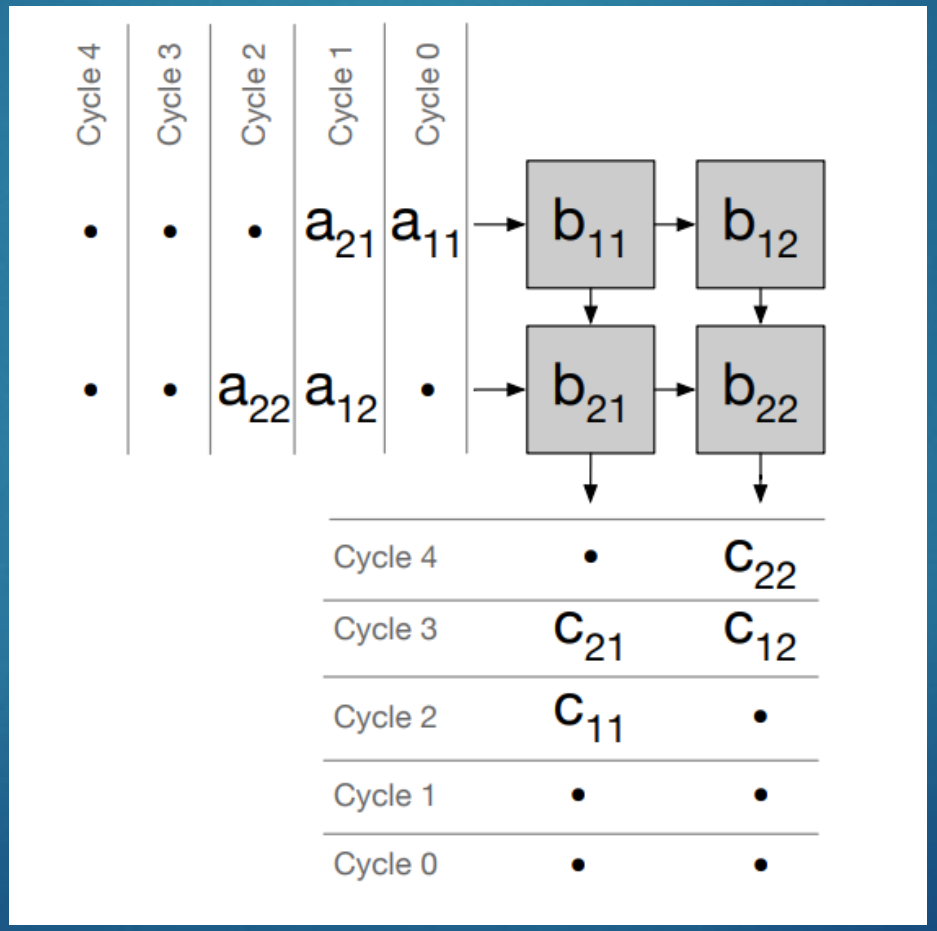
- Forwarded Input Activation (a) to the right
- Forwarded Weight (b) to below
- Partial Sum (cout = a*b + cin)

You are free to add any other IOs to each PE (clk, rst_n, control signals etc as you see fit)

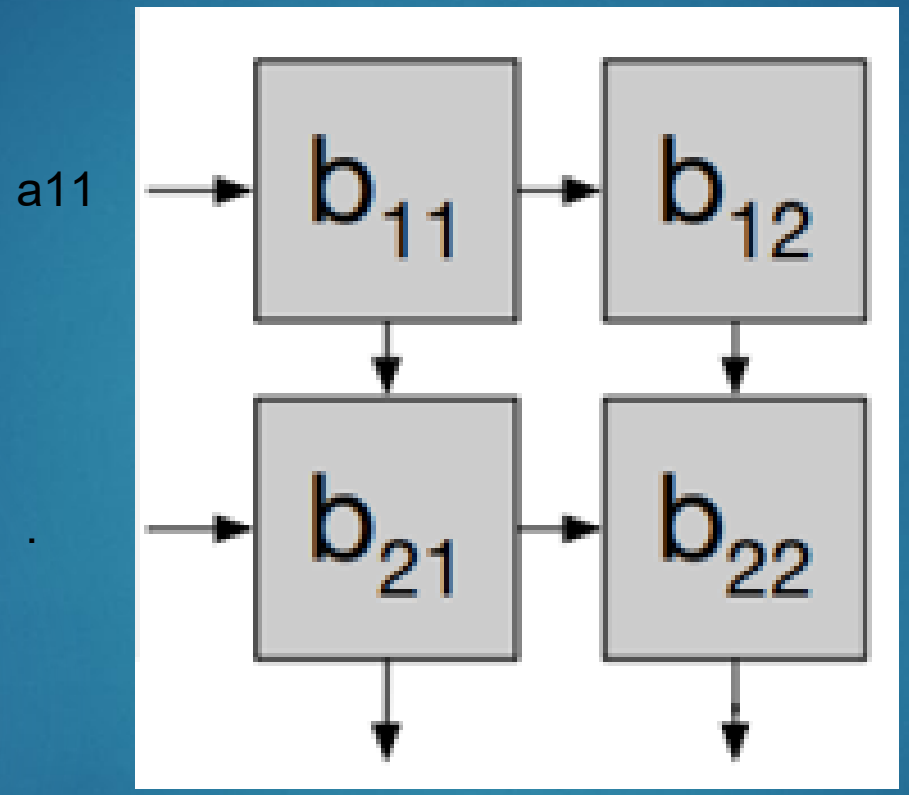


2x2 Systolic Array

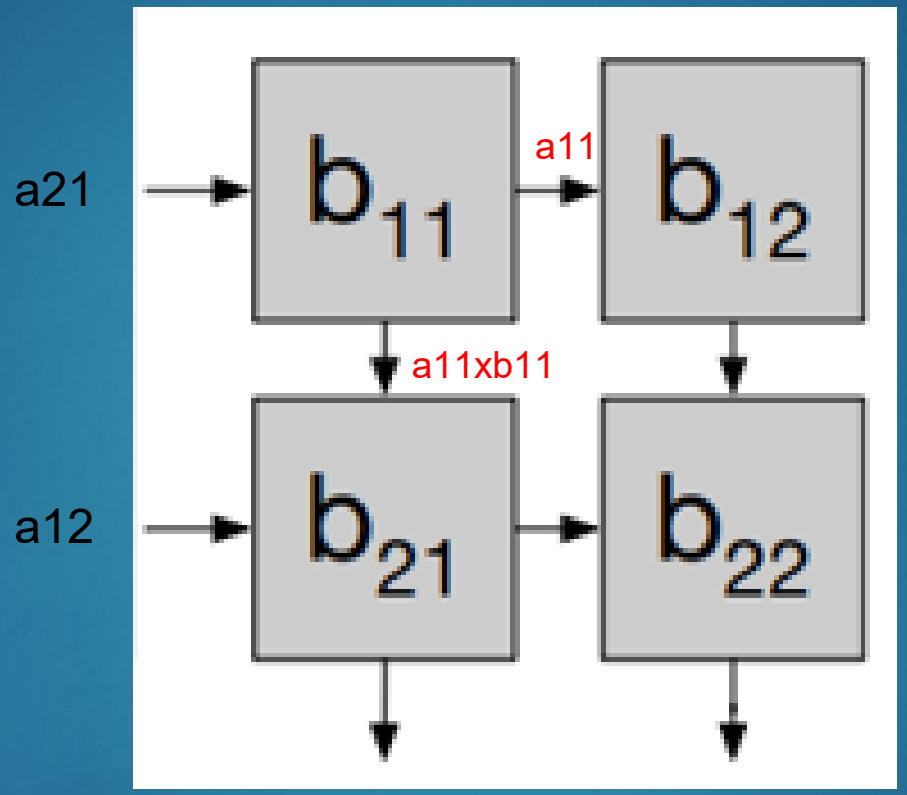
Let us first assume that B matrix is already loaded to the systolic array.
To compute C = AxB, we load A to the systolic array and read out the output C from the bottom.
Note that It takes 5 cycles to complete C = AxB for a 2x2 systolic array



2x2 Systolic Array Example – cycle 0



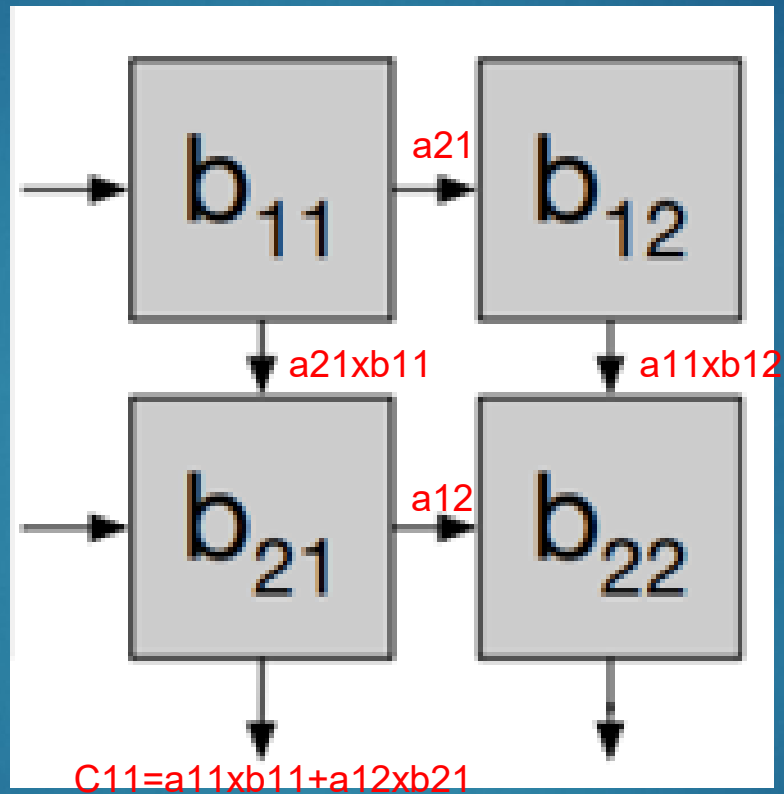
2x2 Systolic Array Example – cycle 1



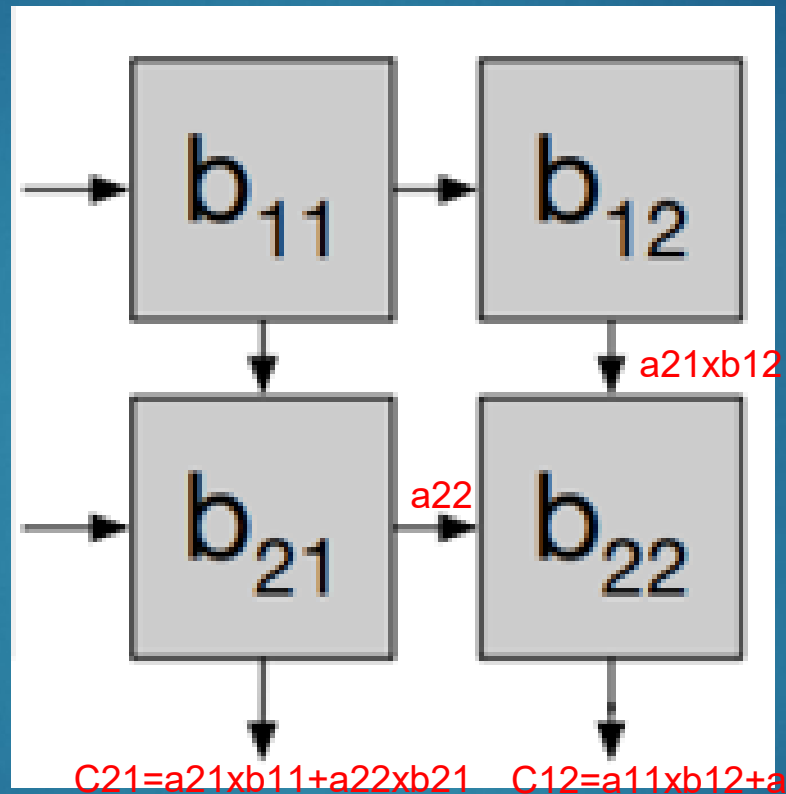
2x2 Systolic Array Example – cycle 2

.

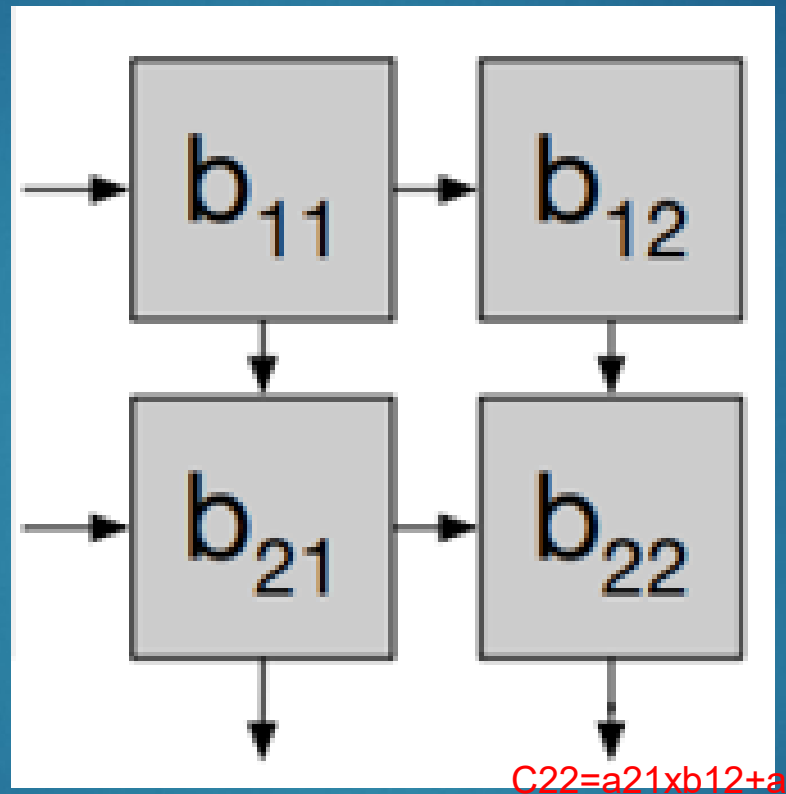
a22



2x2 Systolic Array Example – cycle 3



2x2 Systolic Array Example – cycle 4



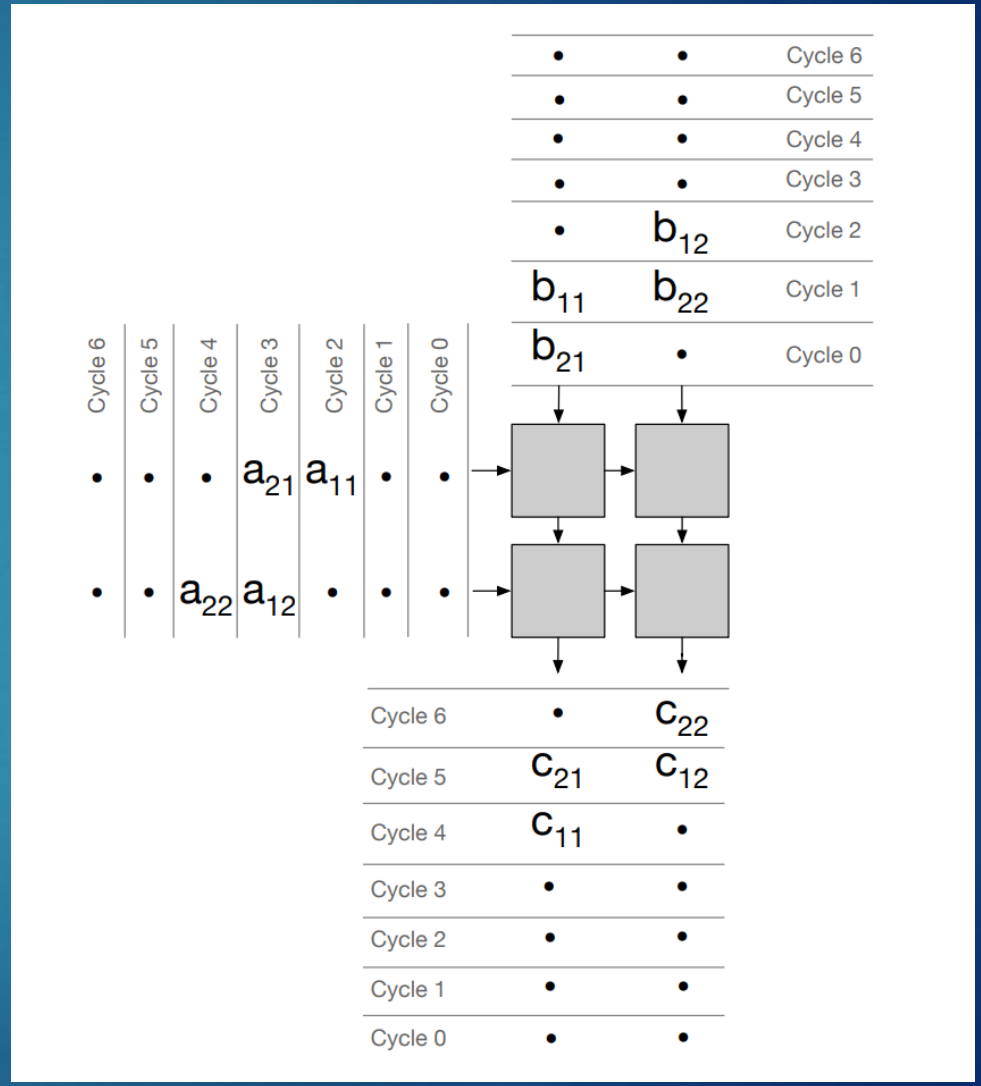
C22=a21xb12+a22xb22

2x2 Systolic Array – Preload Weight

In the above example, we assumed the B matrix is already loaded in the systolic array.

If we want to load B matrix to the array, we need additional 2 cycles of operation

In the example to the right, B matrix is first loaded in to the array (cycle 0 – 2), then the matrix multiplication is performed (cycle 2 – 6)



2x2 Systolic Array with double buffered weight storage

The example above represents an major issue. The systolic array needs to sit idle while the B matrix is loading. If we want to load a new B matrix for each computation, the effective throughput is halved.

To overcome this, we need to **double-buffer** the B matrix storage element in each PE, ie: we need two registers to store B. One register to perform the matrix multiplication for the current cycle; and another register to load the B matrix for the next matrix multiplication.

The example on the right shows a double buffered systolic array performing two consecutive matrix multiply computation.

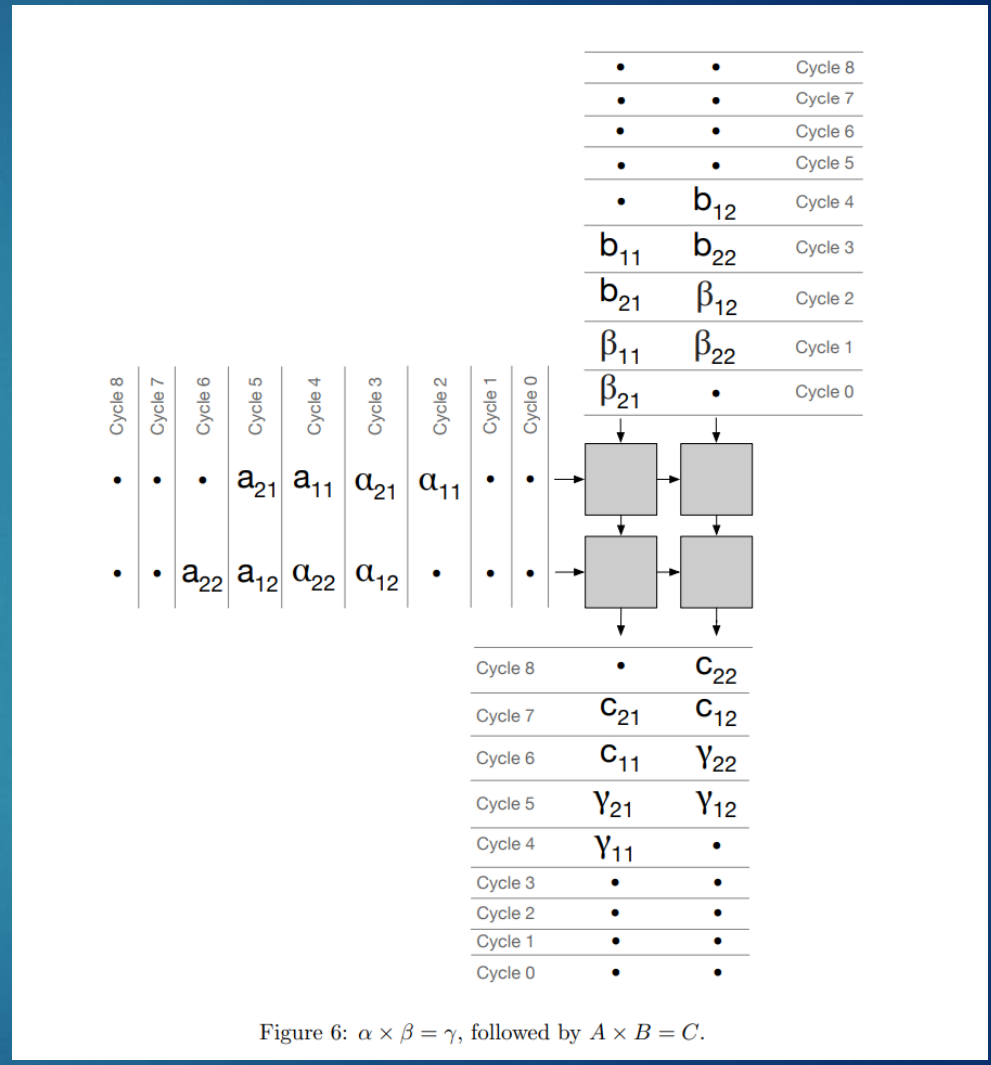


Figure 6: $\alpha \times \beta = \gamma$, followed by $A \times B = C$.

Your assignment

Please write a Verilog code implementing a systolic array controller and a 2x2 systolic array. Then write a testbench to verify the design. Your testbench must include at least two consecutive matrix multiplications as shown in page 11. You may include other test cases in the testbench as you see fit.

Your code should have the following IOs

```
module systolic( … );
  input rst_n, clk; //active low reset and clock
  input [3:0] a11, a12, a21, a22; //input A matrix, assuming each element is 4 bit
  input [3:0] b11, b12, b21, b22; //input B matrix, assuming each element is 4 bit
  input in_valid; //If high, indicates A matrix and B matrix input are valid. If Low they can be ignored
  output [8:0] c11, c12, c21, c22; //output C matrix, assuming each element is 4+4+1=9 bit
  output out_valid; //Describes whether the output is valid. Only  make out_valid high when all 4 elements in the C matrix are valid
```

Your deliverables

Please submit the following to your interviewer:

- RTL code of the systolic array implementation and your testbench to verify the design (you may use Verilog, System Verilog or any other language you are comfortable with for the testbench)
- A short description of your implementation clearly stating any assumptions or other optimizations you made
- If you have a Verilog simulator, then please submit a screenshot of the waveform of two consecutive matrix multiplications as shown in page 11
- If you do not have a Verilog simulator, you are encouraged to at least verify the syntax correctness using an online compiler such as (https://www.tutorialspoint.com/compile_verilog_online.php)

Reference

This project is adapted from: https://inst.eecs.berkeley.edu/~ee290-2/sp20/assets/labs/lab2.pdf
(note you do not need to implement anything in the above link that is not part of this document, eg propagate signal)