# Function

Systolic Array 2x2 utilizes 4 processing nodes/elements (PEs) to perform matrix multiplication and addition before shifting the values to the next PE or output in the chain. For this assignment, the array will have a double weighted buffer to store and prepare the next weight while the current weight is used for matrix multiplication.

To ensure that matrix A values are shifted in sequentially and not read into the PEs for multiplication immediately, shift registers are included in the design. They will shift matrix A values based on the following condition: positive clock edge and when in\_valid is high. This helps to prevent the current cycle’s arithmetic operation from being overwritten by the next one due to blocking assignments used for both multiplication and addition operations.

The weight loading is done using non-blocking assignments to avoid premature placement of values before the current set of matrix multiplications is completed. To prepare the PE for the next set, the next weight is loaded into a buffer register and will be loaded in once the set counter indicates that all current operations are completed.

The outputs of the array are shifted out sequentially to be read out once the last PE in the chain has completed its operation. An output counter is used to indicate the state of the output shifting from the PEs.

For the second PEs in the arithmetic chain (PE1\*), a 1 clock cycle delay is needed to avoid storing unwanted weight values which causes extra wait cycles and incorrect output. A shift\_en signal is added to the PEs to enable the delay if required for the PE. The delay is added if the signal is high, while no delay is added if the signal is low.

# Interface Ports

SystolicArray2x2 –

|  |  |  |
| --- | --- | --- |
| Port Name | Input/Output | Description |
| clk | Input | Central operation clock |
| rstn | Input | Reset active-low signal |
| in\_valid | Input | Indicate if matrices A & B values are valid  Valid if high; Invalid if low |
| out\_valid | Output | Indicate if matrix C values are valid  Valid if high; Invalid if low |
| a00, a01, a10, a11 | Input | Matrix A values  Bus-width configurable by parameter (DATA\_WIDTH) |
| b00, b01, b10, b11 | Input | Matrix B values  Bus-width configurable by parameter (DATA\_WIDTH) |
| c00, c01, c10, c11 | Output | Matrix C values  Bus-width configurable by parameter (ACC\_WIDTH) |

ProcessingElement –

|  |  |  |
| --- | --- | --- |
| Port Name | Input/Output | Description |
| clk | Input | Central operation clock |
| rstn | Input | Reset active-low signal |
| shift\_en | Input | Delay data\_count in PE by 1 cycle if enabled  Enabled if high; disabled if low |
| in\_valid | Input | Indicate if matrices A & B values are valid  Valid if high; Invalid if low |
| out\_valid | Output | Indicate if matrix C values are valid  Valid if high; Invalid if low |
| data\_in | Input | Data value shifted in for matrix multiplication |
| weight\_in | Input | Weight value loaded in for matrix multiplication |
| part\_sum\_in | Input | Output of previous PE shifted in for matrix addition |
| part\_sum\_out | Output | Output of current PE after matrix addition |

# Sequence

A diagram of a cycle

Description automatically generated

Legend –

* Green – Data shifting
* Purple – Weight shifting
* Red – Weight & Product shifting
* Blue – Product (Output) shifting

# Schematics

SystolicArray2x2 –

A computer screen shot of a computer

Description automatically generated

ProcessingElement –

A computer screen shot of a computer circuit

Description automatically generated

# Waveform

All waveform cases have 3 sets of inputs for matrices A and B which produces 3 sets of outputs for matrix C.

Case 1 –

A screenshot of a computer

Description automatically generated

Case 2 –

A screenshot of a computer

Description automatically generated

Case 3 –

A screenshot of a computer

Description automatically generated

Matrix A is 2xN matrix inputs

Matrix B is Nx2 matrix inputs

Matrix calculation within N clock cycles

Consider – 1st PE should take part\_sum\_in as loop back from last (2nd) PE

At N clock cycles,

Update – Use buffer 1 and 2 to do weight storage

Update and choose buffers based on