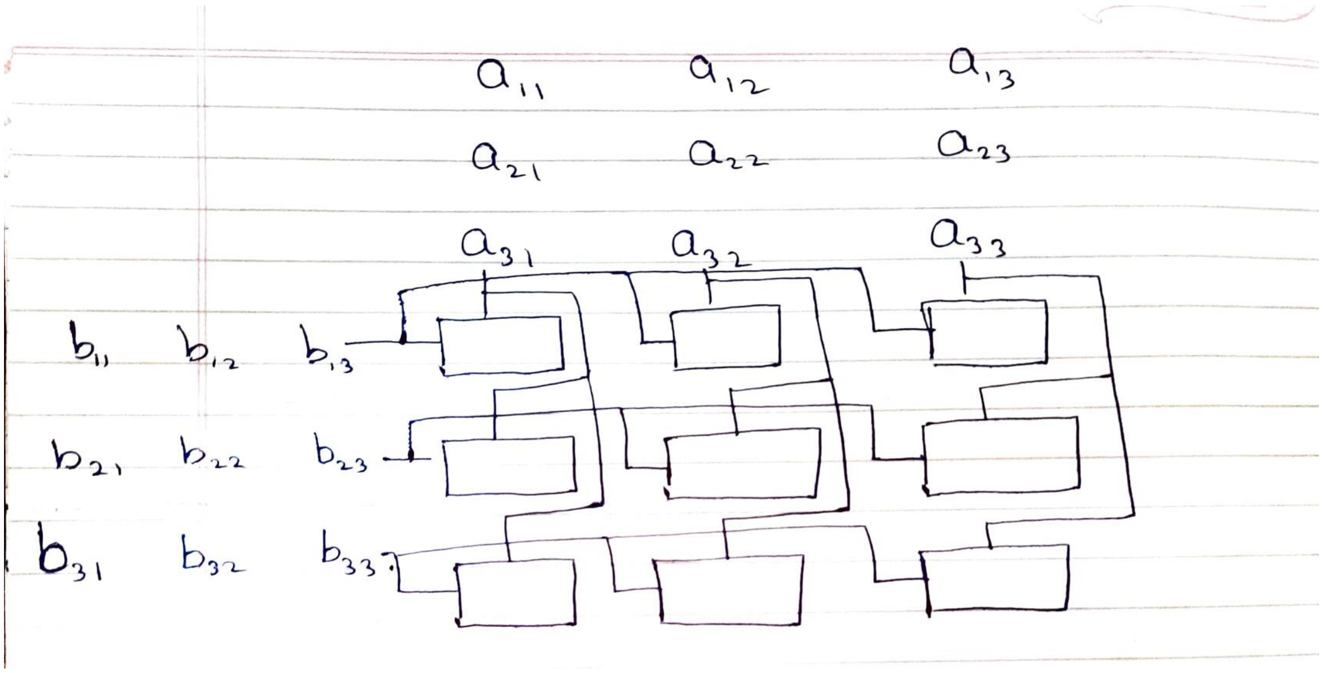
MODIFIED SYSTOLIC ARRAY

The modification I describe involves directly connecting the elements of the input matrices A and B to the processing elements (PEs) in a specific pattern, rather than feeding them into the array in a more sequential manner. This approach aims to reduce the number of clock cycles required to perform the matrix multiplication by exploiting parallel data distribution and processing.

- PE1 is connected to a33 and b33, so it computes a33 × b33
- PE2 is connected to a31 and b32, so it computes a31 × b32
- PE3 is connected to a31 and b31, so it computes a31 × b31
- And so on, following the same pattern.

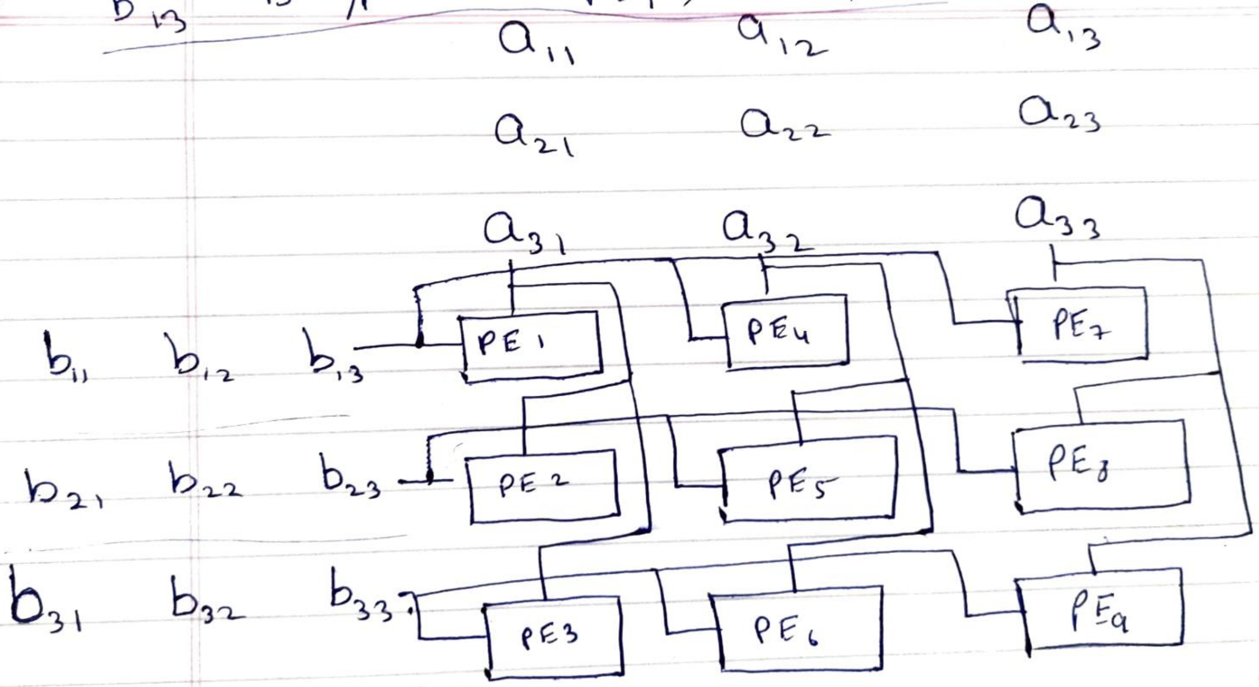
After the first clock cycle in the modified systolic array, the following computations will take place in the processing elements (PEs):

1. PE1 will compute a33 × b33
2. PE2 will compute a31 × b32
3. PE3 will compute a31 × b31
4. PE4 will compute a32 × b33
5. PE5 will compute a32 × b32
6. PE6 will compute a32 × b31
7. PE7 will compute a33 × b33
8. PE8 will compute a33 × b32
9. PE9 will compute a33 × b31

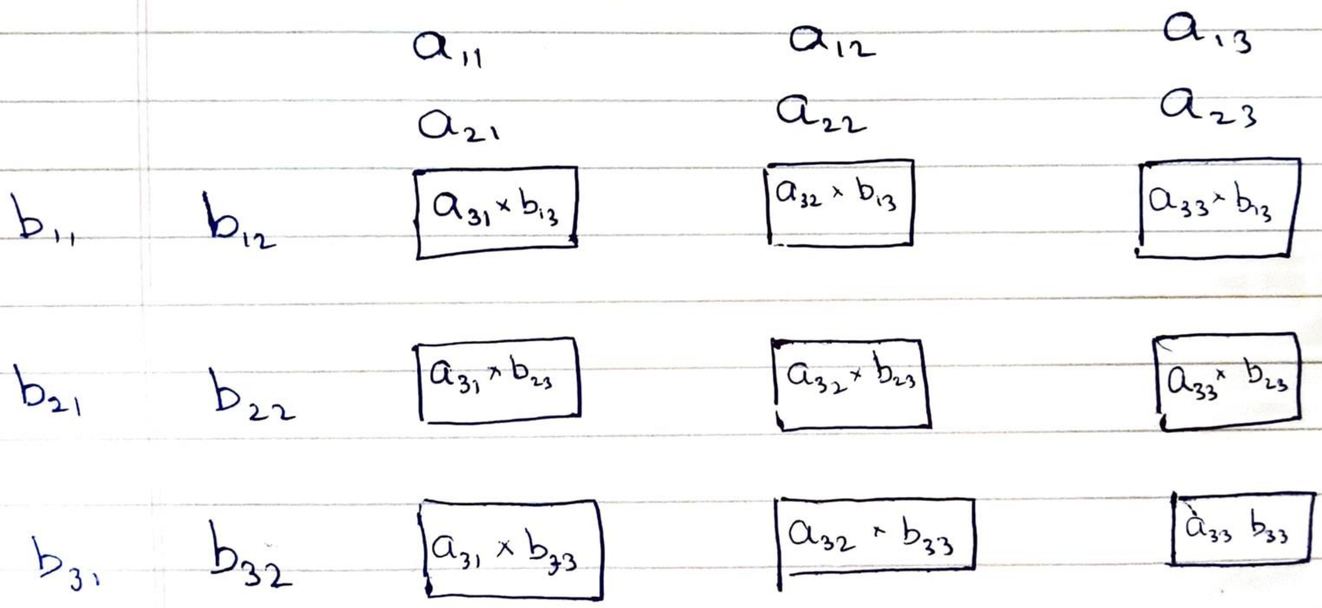


Here a_{31} is i/p to → PE_1, PE_2, PE_3 at same clock edge together

b_{13} is i/p to → PE_1, PE_4, PE_7 → together.

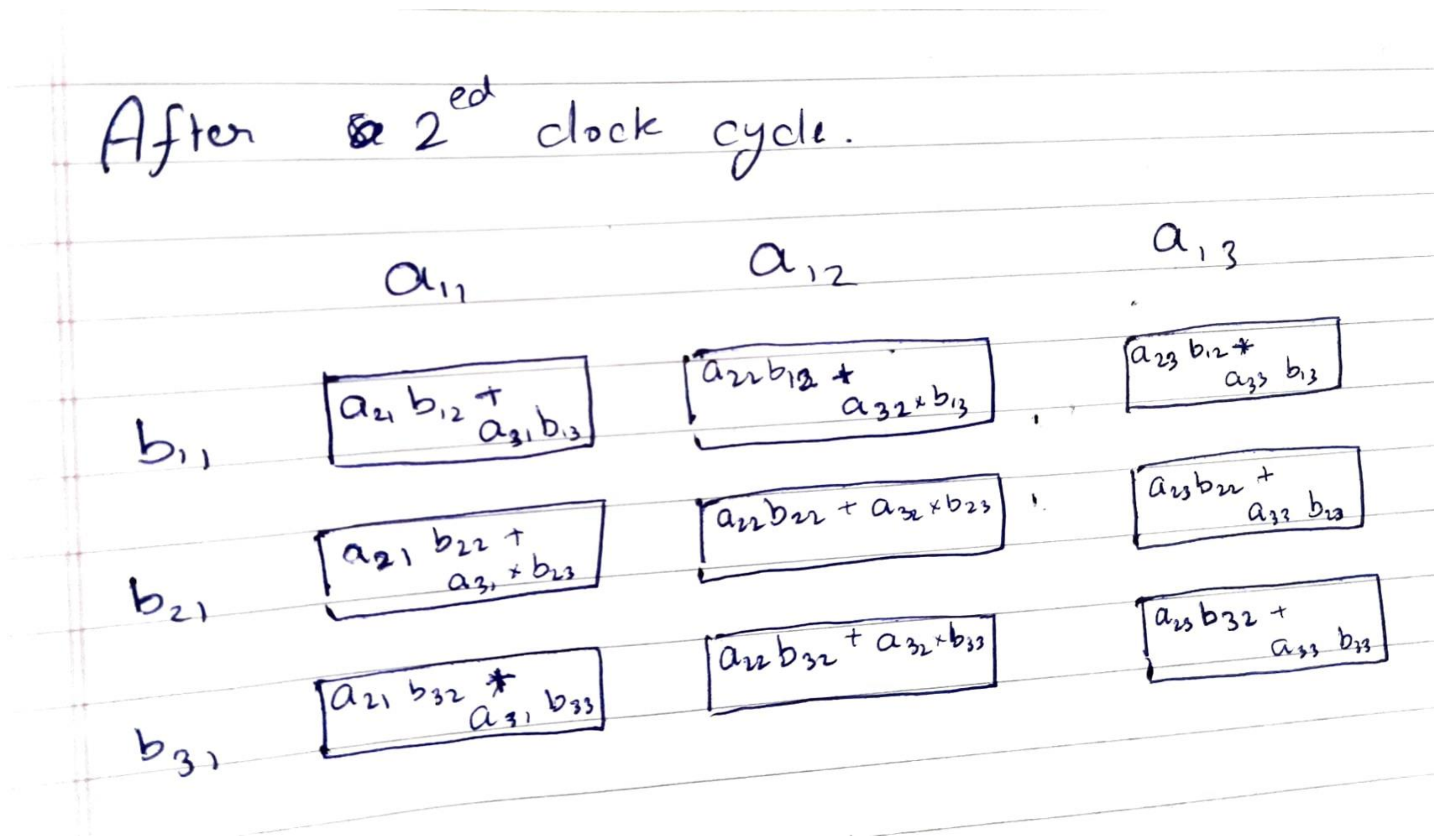


After 1st clock cycle



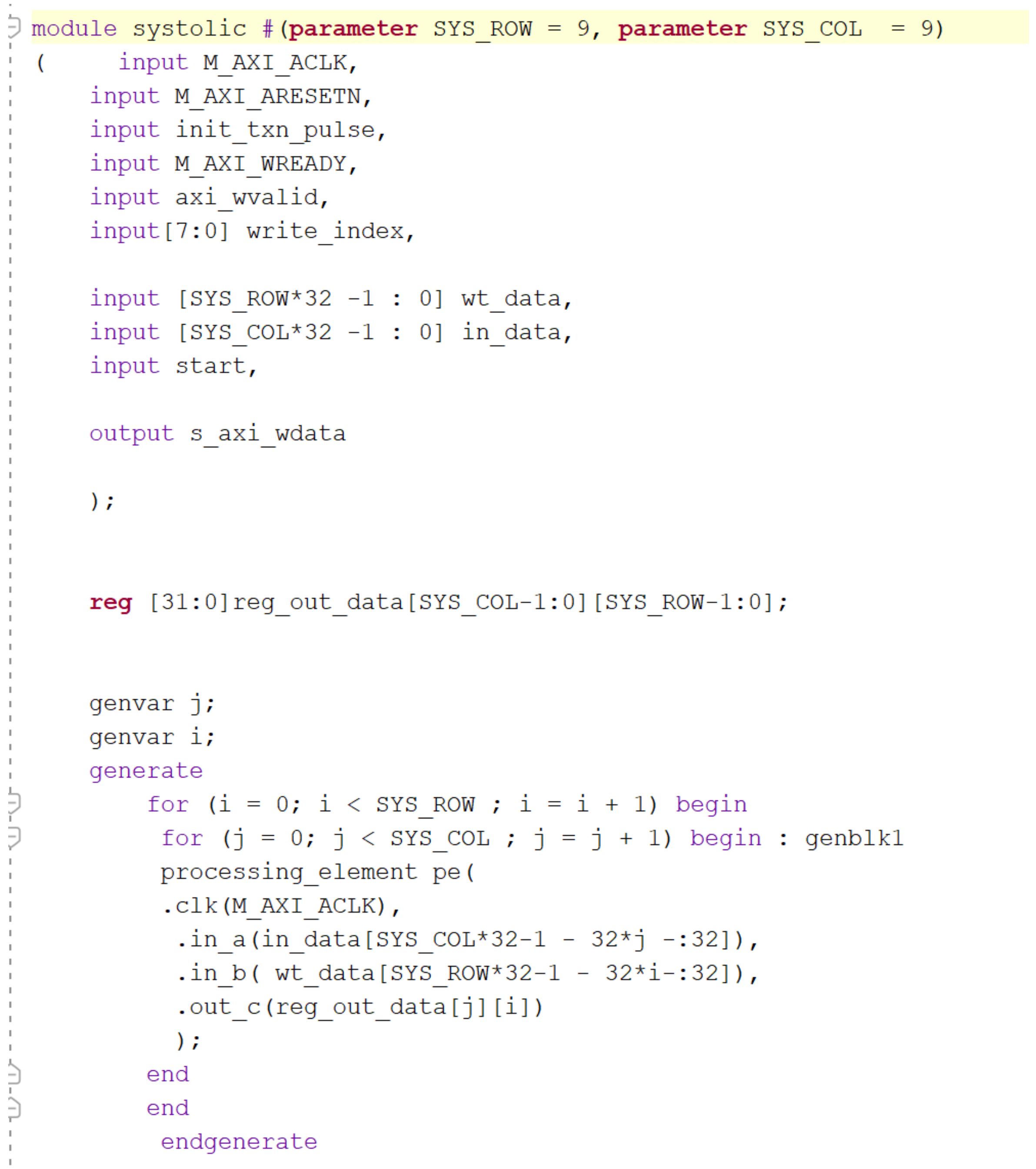
In the second clock cycle of this modified systolic array architecture, the following computations and data propagations will occur:

1. PE1 will accumulate (a21*b12* + *a31*b13)
2. PE2 will accumulate (a21*b22* + *a31*b23)
3. PE3 will accumulate (a21*b32* + *a31*b33)
4. PE4 will accumulate (a22*b12* + *a32*b13).
5. PE5 will accumulate (a22*b22* + *a32*b23).
6. PE6 will accumulate (a22*b32* + *a32*b33).
7. PE7 will accumulate (a23*b12* + *a33*b13)
8. PE8 will accumulate (a23*b22* + *a33*b23).
9. PE9 will accumulate (a23*b32* + *a33*b33).



After ~~2~~ 2ed clock cycle.
a_{11}
a_{12}
a_{13}
b_{11}
b_{21}
b_{31}
$a_{21} b_{12} + a_{31} b_{13}$
$a_{22} b_{12} + a_{32} \times b_{13}$
$a_{23} b_{12} * a_{33} b_{13}$
$a_{21} b_{22} + a_{31} \times b_{23}$
$a_{22} b_{22} + a_{32} \times b_{23}$
$a_{23} b_{22} + a_{33} b_{23}$
$a_{21} b_{32} * a_{31} b_{33}$
$a_{22} b_{32} + a_{32} \times b_{33}$
$a_{23} b_{32} + a_{33} b_{33}$

And so on for 3rd clock cycle.



```
module systolic #(parameter SYS_ROW = 9, parameter SYS_COL  = 9)
(     input M_AXI_ACLK,
    input M_AXI_ARESETN,
    input init_txn_pulse,
    input M_AXI_WREADY,
    input axi_wvalid,
    input[7:0] write_index,

    input [SYS_ROW*32 -1 : 0] wt_data,
    input [SYS_COL*32 -1 : 0] in_data,
    input start,

    output s_axi_wdata

    );


    reg [31:0]reg_out_data[SYS_COL-1:0][SYS_ROW-1:0];


    genvar j;
    genvar i;
    generate
        for (i = 0; i < SYS_ROW ; i = i + 1) begin
         for (j = 0; j < SYS_COL ; j = j + 1) begin : genblk1
         processing_element pe(
         .clk(M_AXI_ACLK),
          .in_a(in_data[SYS_COL*32-1 - 32*j -:32]),
          .in_b( wt_data[SYS_ROW*32-1 - 32*i-:32]),
          .out_c(reg_out_data[j][i])
           );
        end
        end
         endgenerate
```

Inside the genblk1 block, a processing_element instance is created with the following connections:

- clk: Connected to the M_AXI_ACLK input, providing the clock signal.
- in_a: Connected to the corresponding element of the input matrix A (in_data).
- in_b: Connected to the corresponding element of the input matrix B (wt_data).
- out_c: Connected to the corresponding element of the reg_out_data array, which will store the computed result.

The processing_element module (not shown in the provided code) is responsible for performing the actual multiplication and accumulation operations based on the input elements (in_a and in_b). The computed results are stored in the reg_out_data array, which can be accessed through the s_axi_wdata output port.