**1.Problem statement :**

Make a matrix multiplier, where the output C (4x3) is the product of the 2 input matrices A (4x5) and B (5x3). The matrix multiplication is finely pipelined so that in every cycle, the product aij.bjk is produced and added to the partial product cik in location P(i,k). A complete cik is generated in every 5 clock cycles. Assume that the aij, bjk and cik terms are all 32 bit wide integers.

Write VHDL codes of (a) the P(i,k) block, (b) a RAM to read out A, B from and write C into – this RAM has 2 read ports and 1 write port, (c) FIFO (to delay the applications of aij and bjk terms, and delay the writing into the RAM of cik terms

Write a top level VHDL code for the connected arrangement of the P blocks shown in the figure above, and insert appropriate FIFOs for the applications of the aij, bjk and cik terms with correct timings. The RAM holding matrices A, B and C should also be part of this top level design.

**2.Design:**

**High level design:**

**A picture containing sky, screenshot

Description automatically generated**

Matrices A and B are stored in RAM. RAM has 2 read ports and 1 write port. On every clock cycle, the data from RAM is fed into decoder which feeds the data to the matrix multiplier in column wise(matrix B) and row wise(matrix A) manner. The output of matrix multiplier is extracted at the end of matrix multiplication of entire matrix A and B and is then fed to the RAM write port in subsequent clock cycles.

**3. RAM generator and Decoder:**

The RAM generator emulates the RAM which has 2 read ports and 2 write ports. It generates each output in every cycle as follows

a11b11,a21b12,a31b13,a41,a12b21,a22b22,a32b23,a42,a13b31,a23b32,a33b33,a43,a14b41,a24b42,a34b43,a44,a15b51,a25b52,a35b53,a45

This output is fed to the decoder. The decoder feeds this appropriately to the matrix multiplier.

There are 2 decoders, one for matrix A and other for matrix B. Each decoder has 1 input and 4 outputs ( a0,a1,a2,a3 for decoder a and b0,b1,b2,b3 for decoder b). The output A from RAM is the input to decoder A and output B from RAM is input to decoder B.

On every clock cycle, the decoder switches the output from RAM to the appropriate select line e.g., for 1st clock cycle, input a is moved to a0 in decoder a and inout b is moved to b0 in decoder b, thus forming the output element a11 and b11. This process is repeated every clock cycle until end of matrix.

The data is fed into the matrix multiplier which does the matrix multiplication.

**4. Matrix multiplier**

Matrix multiplication is a sequence of multiply, add and accumulate operations. These matrix multiplication operations can be performed in parallel manner. To perform this operation, systolic array has been used in the design.

Systolic array is the heart of the matrix multiplication and is an array of MAC (Multiply Add and accumulate) units as shown in Fig 1.

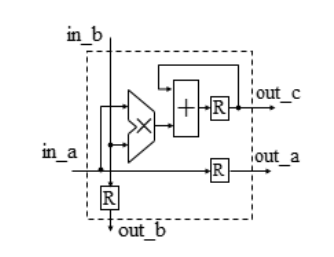
****

Fig 1 : MAC unit/ single PE

Each of such MAC units are called Processing elements. The inputs of each PE is stored in register during the current clock cycle and fed to the following PE as its input in the next clock cycle.

As the multiplication of (4 \* 5) and (5\*3) results in (4\*3) matrix, there would be 12 PEs in the design. The elements of matrix A are fed into the PE in column wise and elements of matrix B are fed into the PE in row wise manner. The data is fed into the PE from the decoder in 20 clock cycles as below

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Clock** | **Elements** | **Clock** | **Elements** | **Clock** | **Elements** | **Clock** | **Elements** |
| #1 | a11b11 | #6 | a22b22 | #11 | a33b33 | #16 | a44 |
| #2 | a21b12 | #7 | a32b23 | #12 | a43 | #17 | a15b51 |
| #3 | a31b13 | #8 | a42 | #13 | a14b41 | #18 | a25b52 |
| #4 | a41 | #9 | a13b31 | #14 | a24b42 | #19 | a35b53 |
| #5 | a12b21 | #10 | a23b32 | #15 | a34b43 | #20 | a45 |

The data flows across the 12 PEs and at the end of 27clock cycle(as in result) the matrix multiplication of 1st matrix is obtained. After the entire matrix has been calculated, the data is fed into RAM every clock cycle. The PEs are then resetand the second matrix is loaded to the systolic array and the calculation is redone.

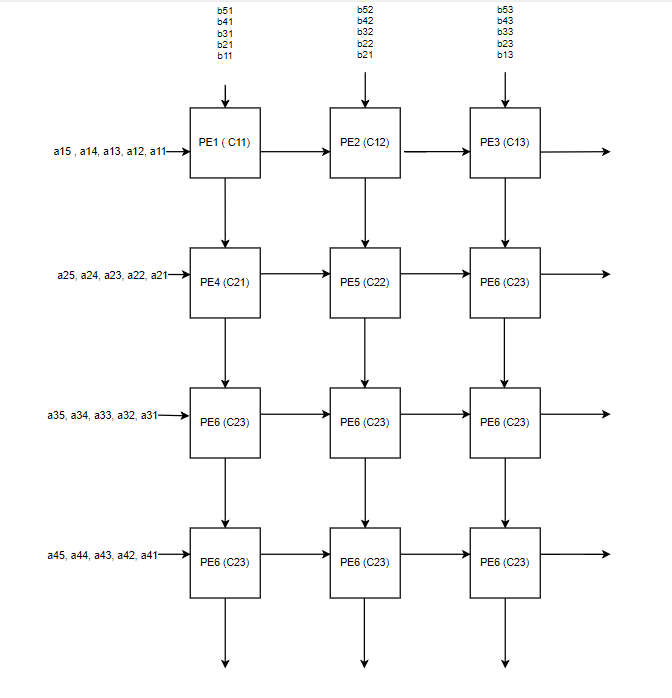


Fig2 : Systolic Array

**Note:** The testbench only provides the clock signals. To view the output the original signals in source code are added to the waveform window.

**3.Source code:**

----Processing element to perform matrix multiplication--

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**USE**ieee.numeric\_std.**ALL**;

**USE**ieee.std\_logic\_unsigned.**ALL**;

**USE**ieee.std\_logic\_arith.**ALL**;

**ENTITY** pe **IS**

**PORT** (

reset :**INOUTstd\_logic** := '0';

clk :**INSTD\_LOGIC**;

ina :**ININTEGERRANGE**0**TO**100;

inb :**ININTEGERRANGE**0**TO**100;

outa, outb, outc :**INOUTINTEGERRANGE**0**TO**100 := 0;

wrt :**OUTINTEGER**);

**END** pe;

**ARCHITECTURE** Behavioral **OF** pe **IS**

**TYPE** RAM **ISARRAY** (0**TO**11) **OFINTEGER**;

**SIGNAL**ram\_a : ram := (**OTHERS** =>0);

**SIGNAL**count :**INTEGER**;

**BEGIN**

**PROCESS** (clk, ina)

**BEGIN**

**IF**rising\_edge(clk) **THEN**

**IF** reset = '1'**THEN**

outa <= 0;

outb<= 0;

outc<= 0;

reset <= '0';

**ELSE**

**IF** count <51**THEN**---restrict to 2 matrix multiplications------

outc<= outc + ina \* inb;

outa <= ina;

outb<= inb;

count <= count + 1;

**ENDIF**;

**ENDIF**;

**ENDIF**;

**ENDPROCESS**;

**END** behavioral;

---------------------------Matrix multiply (top module to do matrix multiplication)----------------

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**USE**IEEE.std\_logic\_unsigned.**ALL**;

**USE**IEEE.std\_logic\_arith.**ALL**;

**ENTITY**matrixmul**IS**

**PORT** (clk :**INSTD\_LOGIC**);

**END**matrixmul;

**ARCHITECTURE** Behavioral **OF**matrixmul**IS**

**COMPONENT**RAMMem**IS**

**PORT** (

rst :**OUTstd\_logic**;

clk :**INSTD\_LOGIC**;

wr :**ININTEGER**;

rdA :**OUTINTEGER** := 0;

rdB :**OUTINTEGER** := 0;

wer :**ININTEGER**

);

**ENDCOMPONENT**;

**COMPONENT** pe **IS**

**PORT** (

reset :**INOUTstd\_logic**;

clk :**INSTD\_LOGIC**;

ina :**ININTEGERRANGE**0**TO**100 := 0;

inb :**ININTEGERRANGE**0**TO**100 := 0;

outa, outb, outc :**INOUTINTEGERRANGE**0**TO**100 := 0;

wrt :**OUTINTEGER**

);

**ENDCOMPONENT**;

**COMPONENT**decodeB**IS**

**PORT** (

clk :**INstd\_logic**;

Dinb :**ININTEGER**;

s0, s1, s2, s3 :**OUTINTEGER** := 0

);

**ENDCOMPONENT**;

**COMPONENT**decodeA**IS**

**PORT** (

clk :**INstd\_logic**;

Dina :**ININTEGER**;

s0, s1, s2, s3 :**OUTINTEGER** := 0

);

**ENDCOMPONENT**;

**SIGNAL**rama, ramb :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL** wr11, wr12, wr13, wr21, wr22, wr23, wr31, wr32, wr33, wr41, wr42, wr43, wers :**INTEGERRANGE**0**TO**100;

**SIGNAL** outa11, outb11, outc11, outa12, outb12, outc12, outa13, outb13, outc13 :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL** outa21, outb21, outc21, outa22, outb22, outc22, outa23, outb23, outc23 :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL** outa31, outb31, outc31, outa32, outb32, outc32, outa33, outb33, outc33 :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL** outa41, outb41, outc41, outa42, outb42, outc42, outa43, outb43, outc43 :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL** a1, a2, a3, a4, b1, b2, b3 :**INTEGERRANGE**0**TO**100 := 0;

**SIGNAL**rstt :**std\_logic** := '0';

**SIGNAL** write, countm :**INTEGER** := 0;

**BEGIN**

-------port map all the components --------------

rAM :RAMMem

**PORTMAP**(rst =>rstt, clk =>clk, wr => write, rdA =>rama, rdb =>ramb, wer =>wers);

deca :decodeA

**PORTMAP**(clk =>clk, Dina =>rama, s0 => a1, s1 => a2, s2 => a3, s3 => a4);

decb :decodeB

**PORTMAP**(clk =>clk, Dinb =>ramb, s0 => b1, s1 => b2, s2 => b3);

pe1 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => a1, inb => b1, outa => outa11, outb => outb11, outc => outc11);

pe2 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa11, inb => b2, outa => outa12, outb => outb12, outc => outc12);

pe3 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa12, inb => b3, outa => outa13, outb => outb13, outc => outc13);

pe4 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => a2, inb => outb11, outa => outa21, outb => outb21, outc => outc21);

pe5 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa21, inb => outb12, outa => outa22, outb => outb22, outc => outc22);

pe6 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa22, inb => outb13, outa => outa23, outb => outb23, outc => outc23);

pe7 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => a3, inb => outb21, outa => outa31, outb => outb31, outc => outc31);

pe8 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa31, inb => outb22, outa => outa32, outb => outb32, outc => outc32);

pe9 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa32, inb => outb23, outa => outa33, outb => outb33, outc => outc33);

pe10 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => a4, inb => outb31, outa => outa41, outb => outb41, outc => outc41);

pe11 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa41, inb => outb32, outa => outa42, outb => outb42, outc => outc42);

pe12 : pe

**PORTMAP**(reset =>rstt, clk =>clk, ina => outa42, inb => outb33, outa => outa43, outb => outb43, outc => outc43);

--------Process to transfer output data to RAM---------------------

**PROCESS** (clk)

**BEGIN**

**IF** (rising\_edge(clk)) **THEN**

countm<= countm + 1;

**IF**countm = 19**OR**countm = 39**THEN**---- o/p value of c after every 19 clk cycles ----

write <= outc11;

**ELSIF**countm = 20**OR**countm = 40**THEN**

write <= outc12;

**ELSIF**countm = 21**OR**countm = 41**THEN**

write <= outc21;

**ELSIF**countm = 22**OR**countm = 42**THEN**

write <= outc13;

**ELSIF**countm = 23**OR**countm = 43**THEN**

write <= outc22;

**ELSIF**countm = 24**OR**countm = 44**THEN**

write <= outc31;

**ELSIF**countm = 25**OR**countm = 45**THEN**

write <= outc23;

**ELSIF**countm = 26**OR**countm = 46**THEN**

write <= outc32;

**ELSIF**countm = 27**OR**countm = 47**THEN**

write <= outc41;

**ELSIF**countm = 28**OR**countm = 48**THEN**

write <= outc33;

**ELSIF**countm = 29**OR**countm = 49**THEN**

write <= outc42;

**ELSIF**countm = 30**OR**countm = 50**THEN**

write <= outc43;

**ENDIF**;

**ENDIF**;

**ENDPROCESS**;

**END** behavioral;

---------------------------RAM generator emulating RAM having 2 read ports and 1 write port-------------------

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**USE**IEEE.std\_logic\_unsigned.**ALL**;

**USE**IEEE.std\_logic\_arith.**ALL**;

**ENTITY**RAMMem**IS**

**PORT** (

rst :**OUTstd\_logic** := '0';

clk :**INSTD\_LOGIC** := '1';

wr :**ININTEGER**;

rdA :**OUTINTEGER** := 0;

rdB :**OUTINTEGER** := 0;

wer :**ININTEGER**

);

**END**RAMMem;

**ARCHITECTURE** Behavioral **OF**RAMMem**IS**

**TYPE**MatA**ISARRAY** (1**TO**4, 1**TO**5) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatB**ISARRAY** (1**TO**5, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatC**ISARRAY** (1**TO**4, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatAA**ISARRAY** (1**TO**4, 1**TO**5) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatBB**ISARRAY** (1**TO**5, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatCC**ISARRAY** (1**TO**4, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatAAA**ISARRAY** (1**TO**4, 1**TO**5) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatBBB**ISARRAY** (1**TO**5, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatCCC**ISARRAY** (1**TO**4, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatAAAA**ISARRAY** (1**TO**4, 1**TO**5) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatBBBB**ISARRAY** (1**TO**5, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

**TYPE**MatCCCC**ISARRAY** (1**TO**4, 1**TO**3) **OFINTEGERRANGE**0**TO**100;

-----1st Matrix A ---------------------

**CONSTANT**matrixA :matA := ((1, 2, 3, 1, 2),

(4, 3, 4, 2, 1), (3, 1, 4, 1, 1),

(1, 2, 3, 5, 1)

);

-----1st Matrix B ---------------------

**CONSTANT**matrixB :matB := ((1, 1, 1),

(1, 1, 1), (1, 1, 1),

(1, 1, 1), (1, 1, 1)

);

**SIGNAL**matrixC :matC := (**OTHERS** => (**OTHERS** =>0));

-----2nd Matrix A ---------------------

**CONSTANT**matrixAA :matAA := ((1, 2, 3, 1, 2),

(2, 3, 1, 2, 1), (3, 1, 2, 1, 1),

(1, 2, 3, 1, 1)

);

-----2nd Matrix A ---------------------

**CONSTANT**matrixBB :matBB := ((1, 1, 1),

(1, 1, 1), (1, 1, 1),

(1, 1, 1), (1, 1, 1)

);

**SIGNAL**matrixCC :matCC := (**OTHERS** => (**OTHERS** =>0));

-----3rd Matrix A ---------------------

**CONSTANT**matrixAAA :matAAA := ((3, 4, 5, 6, 7),

(8, 9, 10, 11, 12),

(13, 14, 15, 16, 17),

(18, 19, 20, 21, 22));

-----3rd Matrix B ---------------------

**CONSTANT**matrixBBB :matBBB := ((23, 24, 25),

(26, 27, 28), (29, 30, 31),

(32, 33, 34), (35, 36, 37));

**SIGNAL**matrixCCC :matCCC := (**OTHERS** => (**OTHERS** =>0));

-----4th Matrix A ---------------------

**CONSTANT**matrixAAAA :matAAAA := ((3, 4, 5, 6, 7),

(8, 9, 10, 11, 12),

(13, 14, 15, 16, 17), (18, 19, 20, 21, 22));

-----4th Matrix B ---------------------

**CONSTANT**matrixBBBB :matBBBB := ((23, 24, 25),

(26, 27, 28), (29, 30, 31),

(32, 33, 34), (35, 36, 37));

**SIGNAL**matrixCCCC :matCCCC := (**OTHERS** => (**OTHERS** =>0));

**SIGNAL**i, j, o, r, state :**INTEGER** := 1;

**SIGNAL**n :**INTEGER** := 2;

**SIGNAL**p :**INTEGER** := 3;

**SIGNAL**q :**INTEGER** := 4;

**SIGNAL** k, l, m :**INTEGER** := 0;

**SIGNAL**count :**INTEGER** := 1;

**SIGNAL** count1, count2, count3, count4, count5, wrcount :**INTEGER** := 0;

**SIGNAL** c11, c12, c13 :**INTEGER** := 0;

**BEGIN**

**PROCESS** (clk)

**BEGIN**

**IF**rising\_edge(clk) **THEN**

**IF**wr /= 0**THEN**

wrcount<= wrcount + 1;

**IF**wrcount = 0**THEN**

c11 <= wr;

**ELSIF**wrcount = 1**THEN**

c12 <= wr;

**ELSIF**wrcount = 2**THEN**

c13 <= wr;

**ENDIF**;

**ENDIF**;

**IF** k = 5**THEN**

rst<= '1';

k <= k + 1;

**ELSE**

------------ Output elements a11b11, a12b21, a13b31,a14b41,a15b51 -------------------------------

**IF** ((count = 1) **OR** count = 5**OR** count = 9**OR** count = 13**OR** count = 17**OR** count = 21**OR** count = 25**OR** count = 29**OR** count = 33**OR** count = 37) **THEN**

**IF** (k <5) **THEN**

rdA<= matrixA(i, j);

rdB<= matrixB(j, i);

i<= i + 1;

**ELSIF** (k <11) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i);

i<= i + 1;

**ELSIF** (k <15) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i);

**ENDIF**;

----------- Output elements a21b12, a22b22, a23b32,a24b42,a25b52 -------------------------------

**ELSIF** ((count = 2) **OR** count = 6**OR** count = 10**OR** count = 14**OR** count = 18**OR** count = 22**OR** count = 26**OR** count = 30**OR** count = 34**OR** count = 38) **THEN**

**IF** (k <5) **THEN**

rdA<= matrixA(i, j);

rdB<= matrixB(j, i);

i<= i + 1;

**ELSIF** (i = 1**AND** j = 1) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i);

i<= i + 1;

**ELSIF** (k <15) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i);

**ENDIF**;

------------ Output elements a31b13, a32b23, a33b33,a34b43,a35b53 -------------------------------

**ELSIF** ((count = 3) **OR** count = 7**OR** count = 11**OR** count = 15**OR** count = 19**OR** count = 23**OR** count = 27**OR** count = 31**OR** count = 35**OR** count = 39) **THEN**

**IF** (k <5) **THEN**

rdA<= matrixA(i, j);

rdB<= matrixB(j, i);

i<= i + 1;

**ELSIF** (i = 1**AND** j = 1) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i);

i<= i + 1;

**ELSIF** (k <15) **THEN**

rdA<= matrixAA(i, j);

rdB<= matrixBB(j, i); **ENDIF**;

------------ Output elements a41b14, a42b24, a43b34,a44b44,a45b54 -------------------------------

**ELSIF** ((count = 4) **OR** count = 8**OR** count = 12**OR** count = 16**OR** count = 20**OR** count = 24**OR** count = 28**OR** count = 32**OR** count = 36**OR** count = 40) **THEN**

**IF** (k <5) **THEN**

rdA<= matrixA(i, j);

**ELSIF** (i = 4**AND** j = 5) **THEN**

rdA<= matrixAA(i, j);

**ELSIF** (k <15) **THEN**

rdA<= matrixAA(i, j);

**ENDIF**;

i<= 1;

IF (k = 4**OR** k = 9**OR** k = 14**OR** k = 19**OR** k = 24**OR** k = 29**OR** k = 34) **THEN**

j <= 1;

l <= l + 1;

**ELSE**

j <= j + 1;

**ENDIF**;

k <= k + 1;

**ENDIF**;

count <= count + 1;

**ENDIF**;

**ENDIF**;

**ENDPROCESS**;

**END** behavioral;

---------Decoder logic to synchronize data transfer(Matrix A elements) between RAM generator and matrix multiplier----------

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**ENTITY**decodeA**IS**

**PORT** (clk :**INstd\_logic**;

Dina :**ININTEGER**;

s0, s1, s2, s3 :**OUTINTEGER** := 0);

**END**decodeA;

**ARCHITECTURE**bhv**OF**decodeA**IS**

**SIGNAL**count :**INTEGER** := 0;

**BEGIN**

**PROCESS** (clk) **IS**

**BEGIN**

**IF**rising\_edge(clk) **THEN**

count <= count + 1;

----Output row wise elements of matrix a for every clk cycle ------------------

**FOR**i**IN**0**TO**30**LOOP**

**IF** (count = 1 + 4 \* i) **THEN**

s0 <= Dina;

s1 <= 0;

s2 <= 0;

s3 <= 0;

**ELSIF** (count = 2 + 4 \* i) **THEN**

s0 <= 0;

s1 <= Dina;

s2 <= 0;

s3 <= 0;

**ELSIF** (count = 3 + 4 \* i) **THEN**

s0 <= 0;

s1 <= 0;

s2 <= Dina;

s3 <= 0;

**ELSIF** (count = 4 + 4 \* i) **THEN**

s0 <= 0;

s1 <= 0;

s2 <= 0;

s3 <= Dina;

**ENDIF**;

**ENDLOOP**;

**ENDIF**;

**ENDPROCESS**;

**END**bhv;

---------Decoder logic to synchronize data transfer(Matrix A elements) between RAM generator and matrix multiplier----------

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**ENTITY**decodeB**IS**

**PORT** (clk :**INstd\_logic**;

Dinb :**ININTEGER**;

s0, s1, s2, s3 :**OUTINTEGER** := 0);

**END**decodeB;

**ARCHITECTURE**bhv**OF**decodeB**IS**

**SIGNAL**count :**INTEGER** := 0;

**BEGIN**

**PROCESS** (clk) **IS**

**BEGIN**

**IF**rising\_edge(clk) **THEN**

count <= count + 1;

----Output row wise elements of matrix b for every clk cycle ------------------

**FOR**i**IN**0**TO**30**LOOP**

**IF** (count = 1 + 4 \* i) **THEN**

s0 <= Dinb;

s1 <= 0;

s2 <= 0;

s3 <= 0;

**ELSIF** (count = 2 + 4 \* i) **THEN**

s0 <= 0;

s1 <= Dinb;

s2 <= 0;

s3 <= 0;

**ELSIF** (count = 3 + 4 \* i) **THEN**

s0 <= 0;

s1 <= 0;

s2 <= Dinb;

s3 <= 0;

**ELSIF** (count = 4 + 4 \* i) **THEN**

s0 <= 0;

s1 <= 0;

s2 <= 0;

s3 <= 0;

**ENDIF**;

**ENDLOOP**;

**ENDIF**;

**ENDPROCESS**;

**4.TestBench Code:**

------------TestBench ----------------------------------

**LIBRARY** IEEE;

**USE** IEEE.STD\_LOGIC\_1164.**ALL**;

**USE**ieee.std\_logic\_unsigned.**ALL**;

**ENTITY**TestBench**IS**

**END**TestBench;

**ARCHITECTURE** Behavioral **OF**TestBench**IS**

**COMPONENT**matrixmul**IS**

**PORT** (clk :**INSTD\_LOGIC**);

**ENDCOMPONENT**;

**COMPONENT**RAMMem**IS**

**PORT** (

clk :**INSTD\_LOGIC**;

wr :**ININTEGER**;

rdA :**OUTINTEGER**;

rdB :**OUTINTEGER**

);

**ENDCOMPONENT**;

**COMPONENT** pe **IS**

**PORT** (

reset :**INstd\_logic**;

clk :**INSTD\_LOGIC**;

ina :**ININTEGERRANGE**0**TO**100;

inb :**ININTEGERRANGE**0**TO**100;

outa, outb, outc :**INOUTINTEGERRANGE**0**TO**100

);

**ENDCOMPONENT**;

**SIGNAL**clk\_tb, rst\_tb :**std\_logic** := '0';

**CONSTANT**clk\_period :**TIME** := 1 ns;

**BEGIN**

uut :matrixmul

**PORTMAP**(clk =>clk\_tb);

**PROCESS**

**BEGIN**

clk\_tb<= '1';

**WAITFOR**clk\_period/2;

clk\_tb<= '0';

**WAITFOR**clk\_period/2;

**ENDPROCESS**;

**6.Elaborated Design:**

As the elaborate design is very large and cannot be accommodated in the report, the snippets are provided.

Below design shows interaction between 12 Processing elements

A close up of a map

Description automatically generated

Below design shows the RAM logic

The RAM interacts with different PE elements via decoders(deca and decb).

