HLS

**5x5**

**#include** <iostream>

**#include** "matrix\_mult\_10x10.h"

**using** **namespace** std;

**int** **main**(**int** argc, **char** \*\*argv)

{

mat\_a in\_mat\_a[5][5] = {

{0, 0, 0, 0, 1 },

{0, 0, 0, 0, 1 },

{0, 0, 0, 0, 1 },

{0, 0, 0, 0, 1 },

{0, 0, 0, 0, 0 }

};

mat\_prod hw\_result, sw\_result;

**int** error\_count = 0;

// Generate the expected result

// Iterate over the rows of the A matrix

// Iterate over the columns of the B matrix

// Do the inner product of a row of A and col of B

Trace: **for**(**int** k = 0; k < IN\_A\_COLS; k++) {

sw\_result+= in\_mat\_a[k][k];

}

**#ifdef** HW\_COSIM

// Run the Vivado HLS matrix multiplier

matrix\_mult(in\_mat\_a, hw\_result);

**#endif**

**#ifdef** HW\_COSIM

// Check result of HLS vs. expected

**if** (hw\_result != sw\_result) {

error\_count++;

**#else**

cout << sw\_result[i][j];

**#endif**

}

**#ifdef** HW\_COSIM

**if** (error\_count)

cout << "TEST FAIL: " << error\_count << "Results do not match!" << **endl**;

**else**

cout << "Test passed!" << **endl**;

**#endif**

**return** error\_count;

}

**#ifndef** \_\_MATRIXMUL\_H\_\_

**#define** \_\_MATRIXMUL\_H\_\_

**#include** <cmath>

**using** **namespace** std;

// Compare TB vs HW C-model and/or RTL

**#define** HW\_COSIM

**#define** IN\_A\_ROWS 5

**#define** IN\_A\_COLS 5

**typedef** **char** mat\_a;

**typedef** **char** mat\_b;

**typedef** **short** mat\_prod;

// Prototype of top level function for C-synthesis

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod);

**#endif** // \_\_MATRIXMUL\_H\_\_ not defined

**#include** "matrix\_mult\_10x10.h"

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod)

{

// Iterate over the rows of the A matrix

prod = 0;

**for**(**int** k = 0; k < 5; k++) {

prod+= a[k][k];

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**10x10**

**#include** "matrix\_mult\_10x10.h"

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod)

{

// Iterate over the rows of the A matrix

prod = 0;

**for**(**int** k = 0; k < 10; k++) {

prod+= a[k][k];

}

}

**#ifndef** \_\_MATRIXMUL\_H\_\_

**#define** \_\_MATRIXMUL\_H\_\_

**#include** <cmath>

**using** **namespace** std;

// Compare TB vs HW C-model and/or RTL

**#define** HW\_COSIM

**#define** IN\_A\_ROWS 10

**#define** IN\_A\_COLS 10

**typedef** **char** mat\_a;

**typedef** **char** mat\_b;

**typedef** **short** mat\_prod;

// Prototype of top level function for C-synthesis

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod);

**#endif** // \_\_MATRIXMUL\_H\_\_ not defined

**#include** <iostream>

**#include** "matrix\_mult\_10x10.h"

**using** **namespace** std;

**int** **main**(**int** argc, **char** \*\*argv)

{

mat\_a in\_mat\_a[10][10] = {

{0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0, 0, 0},

{0, 0, 0, 1, 0, 0, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 0, 0, 0, 0},

{0, 1, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0}

};

mat\_prod hw\_result, sw\_result;

**int** error\_count = 0;

// Generate the expected result

// Iterate over the rows of the A matrix

// Iterate over the columns of the B matrix

// Do the inner product of a row of A and col of B

Trace: **for**(**int** k = 0; k < IN\_A\_COLS; k++) {

sw\_result+= in\_mat\_a[k][k];

}

**#ifdef** HW\_COSIM

// Run the Vivado HLS matrix multiplier

matrix\_mult(in\_mat\_a, hw\_result);

**#endif**

**#ifdef** HW\_COSIM

// Check result of HLS vs. expected

**if** (hw\_result != sw\_result) {

error\_count++;

**#else**

cout << sw\_result[i][j];

**#endif**

}

**#ifdef** HW\_COSIM

**if** (error\_count)

cout << "TEST FAIL: " << error\_count << "Results do not match!" << **endl**;

**else**

cout << "Test passed!" << **endl**;

**#endif**

**return** error\_count;

}

**20X20**

**#include** "matrix\_mult\_20x20.h"

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod)

{

// Iterate over the rows of the A matrix

// Do the inner product of a row of A and col of B

Product: **for**(**int** k = 0; k < IN\_A\_ROWS; k++) {

prod += a[k][k] ;

}

}

**#ifndef** \_\_MATRIXMUL\_H\_\_

**#define** \_\_MATRIXMUL\_H\_\_

**#include** <cmath>

**using** **namespace** std;

// Compare TB vs HW C-model and/or RTL

**#define** HW\_COSIM

**#define** IN\_A\_ROWS 20

**#define** IN\_A\_COLS 20

**typedef** **char** mat\_a;

**typedef** **short** mat\_prod;

// Prototype of top level function for C-synthesis

**void** **matrix\_mult**(

mat\_a a[IN\_A\_ROWS][IN\_A\_COLS],

mat\_prod prod);

**#endif** // \_\_MATRIXMUL\_H\_\_ not defined

**#include** <iostream>

**#include** "matrix\_mult\_20x20.h"

**using** **namespace** std;

**int** **main**(**int** argc, **char** \*\*argv)

{

mat\_a in\_mat\_a[20][20] = {

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

};

mat\_prod hw\_result, sw\_result;

**int** error\_count = 0;

// Generate the expected result

// Iterate over the rows of the A matrix

// Do the inner product of a row of A and col of B

**for**(**int** k = 0; k < IN\_A\_ROWS; k++) {

sw\_result += in\_mat\_a[k][k];

}

**#ifdef** HW\_COSIM

// Run the Vivado HLS matrix multiplier

matrix\_mult(in\_mat\_a, hw\_result);

**#endif**

**#ifdef** HW\_COSIM

// Check result of HLS vs. expected

**if** (hw\_result!= sw\_result) {

error\_count++;

**#else**

cout << sw\_result;

**#endif**

}

**#ifdef** HW\_COSIM

**if** (error\_count)

cout << "TEST FAIL: " << error\_count << "Results do not match!" << **endl**;

**else**

cout << "Test passed!" << **endl**;

**#endif**

**return** error\_count;

}