Matrix Math package for VHDL

A Matrix is just a way of looking at a group of associated numbers.

- Functions similar to those in “Matlab”.
- Based on the built in types “real”, “integer”
- Compatible with VHDL-93, VHDL-2001 and VHDL-2008
- Extended to work with complex numbers
- Extendable to cover synthesizable data types.

This is a proposed package for the VHDL-201X standard.

Packages can be downloaded from: http://www.vhdl.org/fphdl

Type Definitions

Two new types are defined.

```
type real_matrix is array (NATURAL range <>, NATURAL range <>) of REAL;
type integer_matrix is array (NATURAL range <>, NATURAL range <>) of INTEGER;
```

And in the Complex_Matrix package:

```
type complex_matrix is array (NATURAL range <>, NATURAL range <>) of COMPLEX;
type complex_polar_matrix is array (NATURAL range <>, NATURAL range <>) of COMPLEX_POLAR;
```

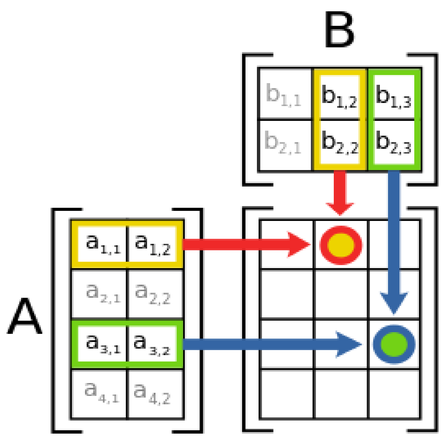
Dependancies

```
Ieee.math_real.all;
Ieee.math_complex.all;
```

Basic matrix math

```
use ieee_proposed.real_matrix_pkg.all;

signal a : real_matrix (1 to 2, 1 to 4);
signal b : real_matrix (1 to 3, 1 to 2);
signal c : real_matrix (1 to 3, 1 to 4);
begin
…
c <= a * b; -- Matrix Multiply
```



Taking a Matrix apart

Take the matrix:

```
variable A : real_matrix (0 to 3, 0 to 3);
A := ((1.0, 2.0, 3.0, 4.0),
      (5.0, 6.0, 7.0, 8.0),
      (9.0, 10.0, 11.0, 12.0),
      (13.0, 14.0, 15.0, 16.0));
B := submatrix (A, 1,1,2,2);
```

Would return a 2x2 matrix (real_matrix (0 to 1, 0 to 1)) starting at location (1,1) in the input matrix A, or:

```
B := ((6.0, 7.0),
      (10.0, 11.0));
```

If the number of rows is “1”, then a vector can be used:

```
Variable BV : real_vector (0 to 2);
BV := submatrix (A, 1,0, 1, 3);
```

Would return 1 row, 3 columns (real_vector (0 to 2)) starting at location (1,0) or:

```
BV := (5.0, 6.0, 7.0);
```

Matrix Inversion

```
use ieee_proposed.real_matrix_pkg.all;

signal a : real_matrix (0 to 2, 0 to 2);
signal b : real_matrix (0 to 2, 0 to 2);
…
b <= inv (a);
```

For a 2x2 Matrix:

$$A^{-1} = \frac{1}{a\,d - b\,c}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

For a 3x3 Matrix:

$$A^{-1} = \frac{1}{|A|}\begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{23} & a_{21} \\ a_{33} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{11} \\ a_{23} & a_{21} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{11} \\ a_{32} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}.$$

Creating a Matrix

```
variable A : real_matrix (0 to 3, 0 to 3);
variable BV, CV : real_vector (0 to 3);
…
BV := (5.0, 6.0, 7.0, 8.0);
CV:= (10.0. 11.0, 12.0, 13.0);
-- Fill Matrix with 1.0
A := ones (A’length(1), A’length(2));
-- Put Vector BV in Matrix A at 2,0 along the “X” (row) axis
BuildMatrix (BV, A, 2, 0);
-- Put Vector CV in Matrix A at 0,2 along the “Y” (column) axis
InsertColumn (CV, A, 0, 2);
```

Will result in:

```
((1.0, 1.0, 10.0, 1.0),
 (1.0, 1.0, 11.0, 1.0),
 (5.0, 6.0, 12.0, 8.0),
 (1.0, 1.0, 13.0, 1.0));
```

Operators

- Matrix multiply
+ Matrix addition
- Matrix subtraction
/ matrix division
** Multiply a matrix by itself. -1 = Matrix inversion
= , /= compare functions
Abs – Absolute value

Arithmetic Functions

Times - Similar to matlab “.*” function (element by element multiply)
Rdivide - Similar to matlab ./ function (element by element divide)
Mrdivide - Similar to matlab mrdivide function (l * inv(r))
Mldivide - Similar to matlab mldivide function (inv(l)* r)
Pow - Similar to matlab “.^” function, (element by element l**r)
Sqrt - element by element square root function
Exp - element by element exp function
Log - element by element natural log function
Trace - Sum the diagonal of a matrix
Sum (matrix, dim) - returns the sum of a matrix along a given x or y
Prod (matrix,dim) - returns the arithmetic multiplication of the input
Dot - returns the dot product of two vectors
Cross - returns the cross product of two matrices
Kron - returns the Kronecker product of two matrices
Det - returns the determinant of a matrix
Inv - Inverts a matrix
Linsolve (matrix, vector) - Solves a linear equation
Normalize(matrix, rval) - Normalizes a matrix
Polyval - Evaluates a polynomial

Functions

Isempty - returns true if the matrix or vector is null
Transpose - Transposes a matrix
Repmat (val) - Creates a matrix by replicating a single value
Zeros - returns a matrix of zeros
Ones - returns a matrix of ones.
Eye - returns an identity matrix
Rand - returns a matrix of random numbers
Cat (dim, l, r) - Concatenates two matrices
Horzcat (l, r) - Concatenates two matrices horizontally
Vertcat (l, r) - Concatenates two matrices vertically
Flipdim (arg, dim) - Flips a matrix along a given dimension
Fliplr - Flip a matrix left to right
Flipup - flip a matrix top to bottom
Rot90 - rotates a matrix 90 degrees
Reshape - reads a matrix and creates one with new dimensions
Size - returns the size of a matrix
Isvector - returns true if the matrix has only one dimension
Isscalar - returns true if there is only one element in this matrix
Numel - returns the number of elements in a matrix
Diag - returns a vector which is the diagonal of a matrix
Blkdiag - returns the block diagonal of a vector.
Blockdiag - Replicates matrix along the diagonal
Repmat - replicates the matrix rows*columns times
Tril - returns the lower triangle of a matrix
Triu - returns the upper triangle of a matrix
Submatrix - returns a submatrix of the given argument
Buildmatrix – Builds a matrix by replicating the input
InsertColum – Inserts a column into a matrix.
Exclude - Return a matrix with the a row or column removed