Library IEEE ;

 Use IEEE . std\_logic\_1164 . All ;

 Use IEEE . numeric\_std . All ;

 Package matrix\_types Is

Type matrix\_2x2 Is Array (1 to 2, 1 to 2) Of

Signed (31 Downto 0); --row x column

Type matrix\_2x1 Is Array (1 to 2) Of

 Signed (31 Downto 0); --row x column

 Type matrix\_1x2 Is Array (1 to 2) Of

 Signed (31 Downto 0); --row x column

End Package matrix\_types ;

 Library IEEE ;

 Use IEEE . std\_logic\_1164 . All ;

 Use IEEE . numeric\_std . All ;

 Use work . matrix\_types . All ;

 Use std. textio . All ;

 Entity kf\_top Is

 Port ( z\_position : In Signed (31 Downto 0);

 reset : In Std\_logic ;

 position , velocity : Out Signed (31 Downto 0));

 End kf\_top ;

 Architecture kf\_behav Of kf\_top Is

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply two 2x2 matrices \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_2x2 (A,B: matrix\_2x2 )

 Return matrix\_2x2 Is

 Variable result : matrix\_2x2 ;

 Variable func\_temp1 : Signed (63 Downto 0) := ( OTHERS => '0');

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For L In 1 to 2 Loop

 For j In 1 to 2 Loop

 func\_temp1 := (A(i,j)\*B(j,L)) + func\_temp1 ;

 End Loop ;

 result (i,L) := func\_temp1 (47 Downto 16) ;

 func\_temp1 :=( OTHERS => '0');

 End Loop ;

 End Loop ;

 Return result ;

 End matrix\_mult\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to add two 2x2 matrices \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_add\_2x2 (A,B: matrix\_2x2 )

 Return matrix\_2x2 Is

 Variable result : matrix\_2x2 ;

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For j In 1 to 2 Loop

 result (i,j) := A(i,j)+B(i,j);

 End Loop ;

 End Loop ;

 Return result ;

 End matrix\_add\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to add a scalar to a 2x2 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_add\_int\_2x2 (A: matrix\_2x2 ;B: Signed (31 Downto 0))

 Return matrix\_2x2 Is

 Variable result : matrix\_2x2 ;

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For j In 1 to 2 Loop

 result (i,j) := A(i,j)+B;

 End Loop ;

 End Loop ;

End matrix\_add\_int\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply a 2x2 with a 2x1 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_2x2\_2x1 (A: matrix\_2x2 ;B: matrix\_2x1 )

 Return matrix\_2x1 Is

 Variable result : matrix\_2x1 ;

 Variable func\_temp1 : Signed (63 Downto 0) := ( OTHERS => '0');

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For j In 1 to 2 Loop

 func\_temp1 := (A(i,j)\*B(j)) + func\_temp1 ;

 End Loop ;

 result (i) := func\_temp1 (47 Downto 16);

 func\_temp1 :=

 ( OTHERS => '0');

 End Loop ;

 Return result ;

 End matrix\_mult\_2x2\_2x1 ;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply a 1x2 with a 2x2 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_1x2\_2x2 (A: matrix\_1x2 ;B: matrix\_2x2 )

 Return matrix\_1x2 Is

 Variable result : matrix\_1x2 ;

 Variable func\_temp1 : Signed (63 Downto 0) :=

 ( OTHERS => '0');

 Begin --Begin function code .

 For L In 1 to 2 Loop

 For j In 1 to 2 Loop

 func\_temp1 := (A(j)\*B(j,L)) + func\_temp1 ;

 End Loop ;

result (L) := func\_temp1 (47 Downto 16);

 func\_temp1 := ( OTHERS => '0');

 End Loop ;

 Return result ;

 End matrix\_mult\_1x2\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply a 1x2 with a 2x1 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_1x2\_2x1 (A: matrix\_1x2 ;B: matrix\_2x1 )

 Return Signed Is

 Variable result : Signed (63 Downto 0) := ( OTHERS => '0');

 Begin --Begin function code .

 For j In 1 to 2 Loop

 result := (A(j)\*B(j)) + result ;

 End Loop ;

 Return result (47 Downto 16);

 End matrix\_mult\_1x2\_2x1 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply a 2x1 and a 1x2 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_2x1\_1x2 (A: matrix\_2x1 ;B: matrix\_1x2 )

 Return matrix\_2x2 Is

 Variable result : matrix\_2x2 ;

 Variable func\_temp1 : Signed (63 Downto 0);

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For L In 1 to 2 Loop

 func\_temp1 := (A(i)\*B(L));

 result (i,L) := func\_temp1 (47 Downto 16) ;

 End Loop ;

Return result ;

 End matrix\_mult\_2x1\_1x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to multiply a 2x1 matrix and a scalar \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_mult\_2x1\_int (A: matrix\_2x1 ;B: Signed (31 Downto 0))

 Return matrix\_2x1 Is

 Variable result : matrix\_2x1 ;

 Variable func\_temp1 : Signed (63 Downto 0);

 Begin --Begin function code .

 For i In 1 to 2 Loop

 func\_temp1 := A(i)\*B;

 result (i) := func\_temp1 (47 Downto 16);

 End Loop ;

 Return result ;

 End matrix\_mult\_2x1\_int ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to add a 2x1 to a 2x1 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_add\_2x1\_2x1 (A: matrix\_2x1 ;B: matrix\_2x1 )

 Return matrix\_2x1 Is

 Variable result : matrix\_2x1 ;

 Begin --Begin function code .

 For i In 1 to 2 Loop

 result (i) := A(i)+B(i);

 End Loop ;

 Return result ;

 End matrix\_add\_2x1\_2x1 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to subtract two 2x2 matrices \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function matrix\_subtract\_2x2 (A,B: matrix\_2x2 )

 Return matrix\_2x2 Is

 Variable result : matrix\_2x2 ;

 Begin --Begin function code .

 For i In 1 to 2 Loop

 For j In 1 to 2 Loop

 result (i,j) := A(i,j)-B(i,j);

 End Loop ;

 End Loop ;

 Return result ;

 End matrix\_subtract\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Function to return the diagonal ( diag ) of a 2x2 matrix \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Function diag\_2x2 (A: matrix\_2x2 )

 Return matrix\_2x1 Is

 Variable result : matrix\_2x1 ;

 Begin --Begin function code .

 For i In 1 to 2 Loop

 result (i) := A(i,i);

 End Loop ;

 Return result ;

 End diag\_2x2 ;

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Begin main process . \*

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Begin

 Process ( z\_position , reset ) Is

 Constant dt : Signed (31 Downto 0) := " 00000000000000000001100110011001 ";

--R is the measurement noise covariance .

Constant R : Signed (31 Downto 0) := " 00000000000010100000000000000000 ";

 --Q is " dynamic noise strength " ( process noise covariance ).

 Constant Q : Signed (31 Downto 0) := " 00000000011001000000000000000000 ";

 --G is the noise injection model .

 --This was intended to be 2 rows , 1 column but is represented as 1 row , 2 columns .

 Constant G : matrix\_2x1 :=

 (" 00000000000000000000000000000000 ",

 " 00000000000000010000000000000000 ");

 --This was intended to be 2 rows , 1 column but is represented as

 --1 row , 2 columns .

 Constant B : matrix\_2x1 :=

 (" 00000000000000000000000000000000 ",

 " 00000000000000010000000000000000 ");

 --This was intended to be 2 rows , 1 column but is represented as

 --1 row , 2 columns .

 Constant Bd : matrix\_2x1 :=

 (" 00000000000000000000000101000111 ",

 " 00000000000000000001100110011001 ");

 Constant H : matrix\_1x2 :=

 (" 00000000000000010000000000000000 ",

 " 00000000000000000000000000000000 ");

Constant H\_prime : matrix\_2x1 :=

(" 00000000000000010000000000000000 ",

" 00000000000000000000000000000000 ");

 Constant F : matrix\_2x2 :=

 ((" 00000000000000000000000000000000 ",

 " 00000000000000010000000000000000 "),

 (" 00000000000000000000000000000000 ",

 " 00000000000000000000000000000000 "));

 Constant phi : matrix\_2x2 :=

 ((" 00000000000000010000000000000000 ",

 " 00000000000000000001100110011001 "),

 (" 00000000000000000000000000000000 ",

 " 00000000000000010000000000000000 "));

 Constant phi\_prime : matrix\_2x2 :=

 ((" 00000000000000010000000000000000 ",

 " 00000000000000000000000000000000 "),

 (" 00000000000000000001100110011001 ",

" 00000000000000010000000000000000 "));

 Constant Qd : matrix\_2x2 :=

 ((" 00000000000000000000100010000110 ",

 " 00000000000000001000000000000000 "),

 (" 00000000000000001000000000000000 ",

 " 00000000000010100000000000000000 "));

 Constant Gd : matrix\_2x2 :=

 ((" 00000000000000010000000000000000 ",

 " 00000000000000000000000000000000 "),

 (" 00000000000000000000000000000000 ",

 " 00000000000000010000000000000000 "));

 -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 --\* Definition of Variables \*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 Variable x : matrix\_2x1 :=

 (" 00000000000000010000000000000000 ",

 " 00000000000000010000000000000000 ");

 Variable P : matrix\_2x2 :=

 ((" 00000000000000000100000000000000 ",

 " 00000000000000000000000000000000 "),

 (" 00000000000000000000000000000000 ",

 " 00000000000000000100000000000000 "));

 Variable A : Signed (31 Downto 0);

 Variable residual : Signed (31 Downto 0);

 Variable K : matrix\_2x1 ;

 Variable K\_temp : Signed (33 Downto 0);

 Begin

 If reset = '1' Then

 x :=

 (" 00000000000000010000000000000000 ",

 " 00000000000000010000000000000000 ");

 P :=

 ((" 00000000000000000100000000000000 ",

" 00000000000000000000000000000000 "),

 (" 00000000000000000000000000000000 ",

 " 00000000000000000100000000000000 "));

 End If;

 x := matrix\_mult\_2x2\_2x1 (phi ,x);

 P := matrix\_add\_2x2 (( matrix\_mult\_2x2 (matrix\_mult\_2x2 (phi ,P),phi\_prime )),Qd);

A := matrix\_mult\_1x2\_2x1 ( matrix\_mult\_1x2\_2x2 (H,P),H\_prime )+R;

--error here, its saying A should be constant or in the power of 2 while dividing..

K\_temp := " 0100000000000000000000000000000000 "/A;

 K := matrix\_mult\_2x1\_int ( matrix\_mult\_2x2\_2x1 (P,H\_prime ), K\_temp (31 Downto 0));

 residual := to\_01 (z\_position ) - matrix\_mult\_1x2\_2x1 (H,x);

 x := matrix\_add\_2x1\_2x1 (x, matrix\_mult\_2x1\_int (K,residual));

 P := matrix\_subtract\_2x2 (P, matrix\_mult\_2x2 (matrix\_mult\_2x1\_1x2 (K,H),P));

 position <= x(1);

 velocity <= x(2);

 End Process ;

 End kf\_behav