// NM_testbench_2019.v

```
// rev 02/22/19
// Copyright 2011-2020 General Vision Holding LLC

/***************** Open Source License Notice ************************

 Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated
documentation files (the "Software"), to deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit
persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the
Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTIONWITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/

`timescale 1 ns / 100 ps // timescale for following modules

module NM500_test();

parameter NAVAIL=576;
parameter MAXVECLENGTH=256;

//Declaration of NeuroMem registers accessed by the Testbench CU
parameter NM_NCR=5'h0;
parameter NM_COMP=5'h1;
parameter NM_LCOMP=5'h2;
parameter NM_DIST=5'h3;
parameter NM_COMPINDEX=5'h3;
parameter NM_CAT=5'h4;
parameter NM_AIF=5'h5;
parameter NM_MINIF=5'h6;
parameter NM_MAXIF=5'h7;
parameter NM_ALLCOMP=5'h8;
parameter NM_ALLCAT=5'h9;
parameter NM_NID=5'hA;
parameter NM_GCR=5'hB;
parameter NM_RESETCHAIN=5'hC;
parameter NM_NSR=5'hD;
parameter NM_FORGET=5'hF;
```

```
parameter NM_NCOUNT=5'hF;

wire ds_io, read_io;
wire id_,unclearn, ready;
pullup (id_), (unclearn),(ready);

reg ds;
reg read;
reg [3:0]register;
reg [15:0]data;
reg [7:0]i=8'h00;

pulldown (ds_io);
assign ds_io=  ds;

pullup (read_io);
assign read_io= read;

wire [3:0]reg_io;
assign reg_io[3:0]= register[3:0];

tri1 [15:0]data_io;
assign data_io[15:0]= read ?16'hZZZZ:data[15:0];

reg [15:0]sm;
reg reset_l;
reg[7:0]error;
reg[7:0]G_error;
reg [15:0]neuronindex;

//-----------------------------------------------------
// G_clock generation
//-----------------------------------------------------
//parameter halfperiod = 27; // half of the clock period
parameter halfperiod = 10; // half of the clock period

reg clock;

always
begin
  #halfperiod; clock <= 1'b0 ;
  #halfperiod; clock <= 1'b1 ;
end
```

```
//-----------------------------------------------------
// Single or multiple NeuroMem chips combination
//-----------------------------------------------------
reg CS_n;
wire dco;
wire dc1;
nm_top nm1(.CS_l(CS_n),.G_CLK(clock), .G_RESET_l(reset_l), .DS(ds_io), .RW_l(read_io), .REG(reg_io),
.DATA(data_io),.RDY(ready),
        .ID_l(id_), .UNC_l(unclearn),.DCI(1'b1), .DCO(dc1));

nm_top nm2(.CS_l(CS_n),.G_CLK(clock), .G_RESET_l(reset_l), .DS(ds_io), .RW_l(read_io), .REG(reg_io),
.DATA(data_io),.RDY(ready),
        .ID_l(id_), .UNC_l(unclearn),.DCI(dc1), .DCO(dco));

//-----------------------------------------------------
// Main test sequence
//-----------------------------------------------------
parameter SM_TESTSR=20;
parameter SM_TESTRAMCHECKER=40;
parameter SM_TESTRAMERROR=50;
parameter SM_TESTRAMBAR=60;
parameter SM_TESTRBF1=80;
parameter SM_TESTRBF2=100;
parameter SM_TESTRBF3=210;
parameter SM_TESTRBF4=240;
parameter SM_TESTRBF5=270;
parameter SM_TESTRBF6=290;
parameter SM_TESTEND=400;

parameter VECT1=16'h87A9;
parameter VECT2=16'h2345;
parameter VECT3=16'hBCDE;
parameter VECT4=16'hA9BD;

reg [15:0]ncount;
reg [15:0]ncount1;
reg [7:0]clusterID;
reg [7:0]vlen;
reg [15:0]iter;
//
// change testbench settings HERE
// vlen
//
initial
begin
```

```
data[15:0]=16'h00;
clock <= 1'b0;
sm[15:0]<=0;
iter[15:0]<=0;
ncount[15:0]=16'h0000; ncount1[15:0]=16'h0000;
error[7:0]<=8'h00; G_error<=8'h00;
vlen[7:0]=4'h04;
//vlen[7:0]=MAXVECLENGTH;
ds<=0; read<=1; register[3:0]<=0;
reset_l <= 1'b0;
// for inter chip test
clusterID[7:0]=8'h00;
end

wire master_cu_ready= ready ;

wire tb_ready= master_cu_ready;
wire [15:0]tb_data_out;
assign tb_data_out= data_io;

always @(negedge clock)
case (sm[15:0])
    0:  begin reset_l<=0;CS_n<=1'b1;
        $display("-------------------------------------");
        sm<=sm+1; end
    1:  begin reset_l<=1; sm<=sm+1;CS_n<=1'b1; end

//---------------------------------------
// Count Neurons
//---------------------------------------
    2: if (tb_ready) begin
        $display("Neuron Count Test");
        tb_write(NM_FORGET,16'h0000); sm<=sm+1; end
    3:  if (tb_ready) begin tb_write(NM_NSR,16'h0010); sm<=sm+1; end
    4:  if (tb_ready) begin tb_write(NM_ALLCAT,16'h0001); sm<=sm+1; end
    5:  if (tb_ready) begin tb_write(NM_RESETCHAIN,16'h0001); ncount[15:0]<=16'h0000; sm<=sm+1; end
    6:  if (tb_ready) begin tb_read(NM_CAT); sm<=sm+1; end
    7:  if (tb_ready)
            if (tb_data_out[15:0]==16'h0001)
            begin ncount[15:0]<=ncount[15:0]+1; sm<=sm-1; end
            else sm<=sm+1;
    8:  if (tb_ready)
        begin
        $display("%d : neurons= %d, expecting= %d",$time,ncount, NAVAIL);
        tb_write(NM_NSR,16'h0000); sm<=sm+1; end
```

```
9:  if (tb_ready) begin tb_write(NM_FORGET,16'h0000); sm<=sm+1; end
// dummy toggle to ensure full test coverage
10: if (tb_ready) begin tb_write(NM_NSR,16'hFFFF); sm<=sm+1; end
11: if (tb_ready) begin tb_write(NM_GCR,16'hFFFF); sm<=sm+1; end
12: if (tb_ready) begin tb_write(NM_NSR,16'h0000); sm<=sm+1; end
13: if (tb_ready) begin tb_write(NM_GCR,16'h0001); sm<=sm+1; end
14: begin
    $display("-----------------------------------");
    sm<=SM_TESTSR;
    end
//-----------------------------------------------------------------
// Test SR mode:
// Write all neurons in SR mode with different ncr, 3 comp, aif and cat
// Reset the chain and verify the ncr, aif and cat
// Note that the comp will be verified in the RAMTEST
//-----------------------------------------------------------------
20: if (tb_ready) begin
    $display("SR Test");
    tb_write(NM_FORGET,16'h0000); error<=0; sm<=sm+1; end
21: if (tb_ready) begin tb_write(NM_NSR,16'h0010); sm<=sm+1; end
22: if (tb_ready) begin
        //if (ncount>32) ncount1<=32; else ncount1<=ncount;
        for (i=0;i<=ncount1;i=i+1) writeSRneuron(i);
        sm<=sm+1; end
23: if (tb_ready) begin tb_write(NM_RESETCHAIN,16'h0000); sm<=sm+1;
    iter[15:0] <= 16'h0000; end
24: if (tb_ready) begin tb_read(NM_NCR);sm<=sm+1; end
25: if (tb_ready) begin
        $display("NCR= %h, expecting %h",tb_data_out[15:0], iter[15:0]);
        if(tb_data_out[15:0]!=iter[15:0])error<=sm;
        sm<=sm+1; end
26: if (tb_ready) begin tb_read(NM_AIF);sm<=sm+1; end
27: if (tb_ready) begin
        $display("AIF= %h, expecting %h",tb_data_out[15:0], iter[15:0]+4);
        if(tb_data_out[15:0]!=iter[15:0]+4)error<=sm;
        sm<=sm+1; end
28: if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
29: if (tb_ready) begin
        $display("CAT= %h, expecting 5",tb_data_out[15:0], iter[15:0]+5);
        if(tb_data_out[15:0]!=iter[15:0]+5)error<=sm;
        sm<=sm+1; end
30: if (iter[15:0]< ncount1[15:0])
    begin iter[15:0]<=iter[15:0]+1; sm<=sm-6; end
    else sm<=sm+1;
```

```
/*
    // read CAT from 1st neuron
    24: if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    25: if (tb_ready) begin
            $display("1st neuron CAT= %h, expecting 5",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0005)error<=sm;
            sm<=sm+1; end
    // read NCR from 2nd neuron
    26: if (tb_ready) begin tb_read(NM_NCR);sm<=sm+1; end
    27: if (tb_ready) begin
            $display("2nd neuron NCR= %h, expecting 1",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0001)error<=sm;
            sm<=sm+1; end
    28:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    // read AIF from 3rd neuron
    29: if (tb_ready) begin tb_read(NM_AIF);sm<=sm+1; end
    30: if (tb_ready) begin
            $display("3rd neuron AIF= %h, expecting 6",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0006 || tb_data_out[15:0]==16'hxxxx)error<=sm;
            sm<=sm+1; end
            */
    31: if (tb_ready) begin tb_write(NM_NSR,16'h0000); sm<=sm+1; end
    32:    if (tb_ready) begin
        $display("%d : Results : Error %d",$time,error);
        $display("-------------------------------------");
        if (error !=0) G_error<=G_error+1;
        sm<=SM_TESTRAMCHECKER; end
//---------------------------------------
// Test RAM Checker:
// Set SR mode
// Write the same vector and category in all neurons
// Set normal mode
// Broadcast the same vector for recognition
// MUST read a distance 0 and the unique category
//---------------------------------------
    40: if (tb_ready)
            begin
            $display("Neurons RAM Checker test");
            tb_write(NM_FORGET,16'h0000);
            error<=0; sm<=sm+1; end
    41: if (tb_ready) begin ramtestNM500(16'hAA55,MAXVECLENGTH,16'h5555); sm<=sm+1; end
    //41: if (tb_ready) begin ramtestCM1K(16'hAA55,MAXVECLENGTH,16'h5555); sm<=sm+1; end
    42: if (tb_ready) begin PutVectCst(16'hAA55,MAXVECLENGTH); sm<=sm+1; end
    43: if (tb_ready) begin
            tb_read(NM_DIST);
```

```
        //$display("SNOOP at %d", $time);
        sm<=sm+1; end
44: if (tb_ready) begin
        //$display("DIST= %h, expecting 0",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'h0000) sm<=SM_TESTRAMERROR; else sm<=sm+1;
        end
45:    if (tb_ready) begin
        tb_read(NM_CAT);
        //$display("SNOOP at %d", $time);
        sm<=sm+1; end
46: if(tb_ready) begin
        //$display("CAT= %h, expecting 5555",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'h5555) error<=sm;
        sm<=sm+1;end
47:    if (tb_ready) begin
        tb_read(NM_DIST);
        //$display("SNOOP at %d", $time);
        sm<=sm+1; end
48: if (tb_ready) begin
        //$display("DIST= %h, expecting xFFFF",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'hFFFF)error<=sm;
        sm<=sm+1;end
49: begin
    $display("%d : Results : Error %d",$time,error);
    $display("-------------------------------------");
    if (error !=0) G_error<=G_error+1;
    sm<=SM_TESTRBF1; end
50: if (tb_ready) begin tb_write(NM_NSR,16'h0010); sm<=sm+1; end
51: if (tb_ready) begin tb_write(NM_RESETCHAIN,16'h0000); neuronindex[15:0]<=16'h0000; sm<=sm+1; end
52: if (tb_ready) begin tb_read(NM_DIST); sm<=sm+1; end
53: if (tb_ready) begin
         if(tb_data_out[15:0]!=16'h0000)
         $display("Bad RAM for neuron %d",neuronindex);
         sm<=sm+1;end
54: if (tb_ready) begin tb_read(NM_CAT); sm<=sm+1; end
55: if(tb_ready)
     begin
         if(tb_data_out[15:0]!=16'hFFFF)
             begin neuronindex<=neuronindex+1; sm<=52; end
         else sm<=SM_TESTEND;
     end
```

```
//-------------------------------------------------------
//Test RBF #1
//Learning of slope vectors with 1st comp shifted by 2
//Enter test vector recognized by neuron 1 and neuron 2
//Interrupt operation with standby mode for 3 cc
//-------------------------------------------------------
    80: if (tb_ready)
        begin
        $display("RBF Test 1");
        tb_write(NM_FORGET,16'h0000); error<=0; sm<=sm+1;
        end
    81: if (tb_ready) begin tb_write(NM_MINIF,16'h0002); sm<=sm+1; end
    82: if (tb_ready)
            begin
            for (i=1;i<4;i=i+1) begin PutVectSlope(i[7:0]<<1,8'h08,{8'h00,i[7:0]});end
            sm<=sm+1;
            end
    83: if (tb_ready ) begin PutVectSlope(8'h03,8'h08,16'h0000);sm<=sm+1; end
    84: if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    85: if (tb_ready) begin
            $display("DIST= %h, expecting 8",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0008)error<=sm;
            sm<=sm+1; end
    86: if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    87: if (tb_ready) begin
            $display("CAT= %h, expecting 1",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0001)error<=sm;
            sm<=sm+1; end
    88: if (tb_ready ) begin tb_read(NM_DIST);sm<=sm+1; end
    89: if (tb_ready) begin
            $display("DIST= %h, expecting 8",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0008)error<=sm;
            sm<=sm+1; end
    90: if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    91: if (tb_ready) begin
            $display("CAT= %h, expecting 2",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0002)error<=sm;
            sm<=sm+1; end
    92: if (tb_ready ) begin PutVectSlope(8'h02,8'h08,16'h0000);sm<=sm+1; end
    93:     if (tb_ready) begin tb_read(NM_NSR); sm<=sm+1; end
    94:     if (tb_ready) begin
            $display("CSR= %h, expecting identified status 8",tb_data_out[15:0]);
            if (tb_data_out[15:0]!=16'h0008) error<=sm;
            sm<=sm+1; end
    95:     begin
```

```
            $display("%d : Results: Error %d",$time,error);
            $display("-------------------------------------");
            if (error !=0) G_error<=G_error+1;
            CS_n<=1'b1; sm<=sm+1; end
    96:     sm<=sm+1;
    97:     sm<=sm+1;
    98:     begin CS_n<=1'b0; sm<=SM_TESTRBF2; end

//---------------------------------------
//Test RBF  #2
//Learning and recognition triggering the three
// possible recognition status: ID, UNC, UNK
//
//LEARN     Vector1     Vector2    Vector3
//Context   1           1           1
//Comp      87A9        2345        BCDE
//Cat       11          0           13
//
//RECO      Vector4
//Context   1
//Comp      A9BD
//Dist      9                       7
//---------------------------------------
    100:    if (tb_ready)
            begin
            $display("RBF Test 2");
            tb_write(NM_FORGET,{16'h0000}); error<=0; sm<=sm+1;
            end
    101:    if (tb_ready) begin PutVect4(VECT1,8'h1); sm<=sm+1; end
    102:    if (tb_ready) begin tb_write(NM_CAT,16'h0011); sm<=sm+1; end
    103:    if (tb_ready) begin PutVect4(VECT2,8'h1); sm<=sm+1; end
    104:    if (tb_ready) begin tb_write(NM_CAT,16'h0000); sm<=sm+1; end
    105:    if (tb_ready) begin PutVect4(VECT3,8'h1); sm<=sm+1; end
    106:    if (tb_ready) begin tb_write(NM_CAT,16'h0013); sm<=sm+1; end
    107:    if (tb_ready) begin PutVect4(VECT4,8'h1); sm<=sm+1; end
    108:    if (tb_ready) begin tb_read(NM_NSR); sm<=sm+1; end
    109:    if (tb_ready) begin
            $display("CSR= %h, expecting uncertain status 4",tb_data_out[15:0]);
            if (tb_data_out[15:0]!=16'h0004) error<=sm;
            sm<=sm+1; end
    110:    if (tb_ready) begin tb_read(NM_FORGET);sm<=sm+1; end
    111:    if (tb_ready) begin
            $display("NCOUNT= %h, expecting 2 neurons",tb_data_out[15:0]);
            if (tb_data_out[15:0]!=16'h0002) error<=sm;
            sm<=sm+1; end
```

```
    112:    begin
        $display("%d : Results: Error %d",$time,error);
        $display("-------------------------------------");
        if (error !=0) G_error<=G_error+1;
        //sm<=SM_TESTEND; end
        sm<=SM_TESTRBF3; end
```

```
//----------------------------------------
//Test RBF #3
// Similar to Test RBF1 but using multiple contexts
//
//LEARN     Vector1     Vector2    Vector3
//Context   0x55        0x33        0xAA
//Comp      87A9        2345        BCDE
//Cat       11          12          13
//
//RECO      Vector4
//Context   0x55
//Comp      A9BD
//Dist      9           na          na
//Cat       11

//RECO      Vector4
//Context   0xAA
//Comp      A9BD
//Dist      na           na          7
//CAT       13

//RECO      Vector4
//Context   0
//Comp      A9BD
//Dist      9            na          7
//Cat       11                 na              13
//----------------------------------------
    210:    if (tb_ready)
```

```
        begin
        $display("RBF Test 3 with mixed context values");
        tb_write(NM_FORGET,{16'h0000}); error<=0; sm<=sm+1;
        end
211:    if (tb_ready) begin PutVect4(VECT1,8'h55); sm<=sm+1; end
212:    if (tb_ready) begin tb_write(NM_CAT,16'h0011); sm<=sm+1; end
213:    if (tb_ready) begin PutVect4(VECT2,8'h33); sm<=sm+1; end
214:    if (tb_ready) begin tb_write(NM_CAT,16'h0012); sm<=sm+1; end
215:    if (tb_ready) begin PutVect4(VECT3,8'hAA); sm<=sm+1; end
216:    if (tb_ready) begin tb_write(NM_CAT,16'h0013); sm<=sm+1; end
217:    if (tb_ready) begin PutVect4(VECT4,8'h55); sm<=sm+1; end
218:    if (tb_ready) begin tb_read(NM_NSR); sm<=sm+1; end
219:    if (tb_ready) begin
        $display("CSR= %h, expecting identified status 8",tb_data_out[15:0]);
        if (tb_data_out[15:0]!=16'h0008) error<=sm;
        sm<=sm+1; end
220:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
221:    if (tb_ready) begin
        $display("CAT= %h, expecting 11",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'h0011)error<=sm;
        sm<=sm+1; end
222:    if (tb_ready) begin PutVect4(VECT4,8'hAA); sm<=sm+1; end
223:    if (tb_ready) begin tb_read(NM_NSR); sm<=sm+1; end
224:    if (tb_ready) begin
        $display("CSR= %h, expecting identified status 8",tb_data_out[15:0]);
        if (tb_data_out[15:0]!=16'h0008) error<=sm;
        sm<=sm+1; end
225:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
226:    if (tb_ready) begin
        $display("CAT= %h, expecting 13",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'h0013)error<=sm;
        sm<=sm+1; end
227:    if (tb_ready) begin PutVect4(VECT4,8'h0); sm<=sm+1; end
228:    if (tb_ready) begin tb_read(NM_NSR); sm<=sm+1; end
229:    if (tb_ready) begin
        $display("CSR= %h, expecting uncertain status 4",tb_data_out[15:0]);
        if (tb_data_out[15:0]!=16'h0004) error<=sm;
        sm<=sm+1; end
230:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
231:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
232:    if (tb_ready) begin
        $display("CAT= %h, expecting 13",tb_data_out[15:0]);
        if(tb_data_out[15:0]!=16'h0013)error<=sm;
        sm<=sm+1; end
233:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
```

```
    234:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    235:    if (tb_ready) begin
            $display("CAT= %h, expecting 11",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0011)error<=sm;
            sm<=sm+1; end
    236:    begin
            $display("%d : Results: Error %d",$time,error);
            $display("-----------------------------------");
            if (error !=0) G_error<=G_error+1;
            sm<=SM_TESTRBF4; end

//---------------------------------------
//Test RBF #4
// Similar to Test RBF1 but higher MINIF value
// to degenerate neurons
//
//LEARN     Vector1     Vector2    Vector3
//Context   1           1           1
//Comp      87A9        2345        BCDE
//Cat       11          12          13
//
//RECO      Vector4
//Context   1
//Comp      A9BD
//Dist      9           na          na
//Cat       8013                        8011
//---------------------------------------
    240:    if (tb_ready)
            begin
            $display("RBF Test 4, testing degenerated flags");
            tb_write(NM_FORGET,{16'h0000}); error<=0; sm<=sm+1;
            end
    241:    if (tb_ready) begin tb_write(NM_MINIF, 16'h0012); sm<=sm+1; end
    242:    if (tb_ready) begin PutVect4(VECT1,8'h1); sm<=sm+1; end
    243:    if (tb_ready) begin tb_write(NM_CAT,16'h0011); sm<=sm+1; end
    244:    if (tb_ready) begin PutVect4(VECT2,8'h1); sm<=sm+1; end
    245:    if (tb_ready) begin tb_write(NM_CAT,16'h0012); sm<=sm+1; end
    246:    if (tb_ready) begin PutVect4(VECT3,8'h1); sm<=sm+1; end
    247:    if (tb_ready) begin tb_write(NM_CAT,16'h0013); sm<=sm+1; end
    248:    if (tb_ready) begin PutVect4(VECT4,8'h1); sm<=sm+1; end
    249:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    250:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    251:    if (tb_ready) begin
            $display("CAT= %h, expecting 8013",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h8013)error<=sm;
```

```
            sm<=sm+1; end
    252:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    253:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    254:    if (tb_ready) begin
            $display("CAT= %h, expecting 8011",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h8011)error<=sm;
            sm<=sm+1; end
    255:    if (tb_ready) begin PutVect4(VECT2,8'h1); sm<=sm+1; end
    256:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    257:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    258:    if (tb_ready) begin
            $display("CAT= %h, expecting 0012",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0012)error<=sm;
            sm<=sm+1; end
    259:    if (tb_ready) begin tb_write(NM_MINIF, 16'h0008); sm<=sm+1; end
    260:    if (tb_ready) begin PutVect4(16'h3456,8'h1); sm<=sm+1; end
    261:    if (tb_ready) begin tb_write(NM_CAT,16'h0000); sm<=sm+1; end
    262:    if (tb_ready) begin PutVect4(VECT2,8'h1); sm<=sm+1; end
    263:    if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    264:    if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    265:    if (tb_ready) begin
            $display("CAT= %h, expecting 8012",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h8012)error<=sm;
            sm<=sm+1; end
    266:    begin
            $display("%d : Results: Error %d",$time,error);
            $display("-------------------------------------");
            if (error !=0) G_error<=G_error+1;
            //sm<=SM_TESTRBF2; end
            sm<=SM_TESTRBF5; end
//---------------------------------------
// Test RBF #5
// commit all clusters with the same content:
//        neuron1= context 1, slope vector, category 0x32
//        neuron 2 to 12= context 99, slope vector, category 0x0032
// broadcast context 1 and slope vector
//        1st neuron of each cluster must fire
//        verify nsr = 8
//        verify dist =0, cat = 0x32
//---------------------------------------
    270: if (tb_ready)
        begin
        $display("RBF Test 5");
        tb_write(NM_FORGET,16'h0000); clusterID=0; error<=0; sm<=sm+1;
        end
```

```
    271: if (tb_ready) begin tb_write(NM_NSR,16'h0010); sm<=sm+1; end
    272: if (tb_ready) begin
                  fillclusterSR(8'h01, vlen,16'h0032);
                  $display("Cluster ID= %d",clusterID);
                  clusterID <= clusterID+1;
                  //if (clusterID >=48) sm<=sm+1; end
                  if (clusterID >=(ncount/12)) sm<=sm+1; end
    273: if (tb_ready) begin tb_write(NM_NSR,16'h0000);sm<=sm+1; end
        // verify that network is full
    274: if (tb_ready) begin tb_read(NM_NCOUNT);sm<=sm+1; end
    275: if (tb_ready) begin
            $display("NCOUNT= %h, expecting 0xFFFF",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'hFFFF)error<=sm;
            sm<=sm+1; end
            // broadcast the "firing" slope vector with category null
    276: if (tb_ready) begin tb_write(NM_GCR,16'h0001); sm<=sm+1; end
    277: if (tb_ready) begin PutVectSlope(8'h00,vlen,16'h0000);sm<=sm+1;end
            // verify that the NSR is identified
    278: if (tb_ready ) begin tb_read(NM_NSR);sm<=sm+1; end
    279: if (tb_ready) begin
            $display("NSR= %h, expecting 8",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0008)error<=sm;
            sm<=sm+1; end
    280: if (tb_ready) begin tb_read(NM_DIST);sm<=sm+1; end
    281: if (tb_ready) begin
            $display("DIST= %h, expecting 0",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0000)error<=sm;
            sm<=sm+1; end
    282: if (tb_ready) begin tb_read(NM_CAT);sm<=sm+1; end
    283: if (tb_ready) begin
            $display("CAT= %h, expecting 0x0032",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0032)error<=sm;
            sm<=sm+1; end
    284:     begin
            $display("%d : Results: Error %d",$time,error);
            $display("-------------------------------------");
            if (error !=0) G_error<=G_error+1;
            CS_n<=1'b1; sm<=sm+1; end
    285:    begin CS_n<=1'b0; sm<=SM_TESTRBF6; end

//---------------------------------------
// Test RBF #6
// commit all clusters with the same vectors:
//        neuron1= context 1, slope vector of MAXVECLENGTH components, category 0x0064
//        neuron 2 to 12= context 99, slope vector, category 0x0032
```

```
// broadcast context 1 and slope vector
//        1st neuron of each cluster must fire
//        verify nsr = 4
//---------------------------------------
    290: if (tb_ready)
        begin
        $display("RBF Test 6");
        tb_write(NM_FORGET,16'h0000); clusterID<=0; error<=0; sm<=sm+1;
        end
    291: if (tb_ready) begin tb_write(NM_NSR,16'h0010); sm<=sm+1; end
    292: if (tb_ready) begin
                  fillclusterSR(8'h01, vlen, clusterID+1); // no degenerated neurons thx to SR mode
                  $display("Cluster ID= %d",clusterID);
                  clusterID <= clusterID+1;
                  if (clusterID >=(ncount/12)) sm<=sm+1; end
    293: if (tb_ready) begin tb_write(NM_NSR,16'h0000);sm<=sm+1; end
        // verify that network is full
    294: if (tb_ready) begin tb_read(NM_NCOUNT);sm<=sm+1; end
    295: if (tb_ready) begin
            $display("NCOUNT= %h, expecting 0xFFFF",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'hFFFF)error<=sm;
            sm<=sm+1; end
            // broadcast the "firing" slope vector with category null
    296: if (tb_ready) begin tb_write(NM_GCR,16'h0001); sm<=sm+1; end
    297: if (tb_ready) begin PutVectSlope(8'h00,vlen,16'h0000);sm<=sm+1;end
            // verify that the NSR is identified
    298: if (tb_ready ) begin tb_read(NM_NSR);sm<=sm+1; end
    299: if (tb_ready) begin
            $display("NSR= %h, expecting 4",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0004)error<=sm;
                  clusterID<=0;
            sm<=sm+1; end
    300: if (tb_ready) begin tb_read(NM_DIST); sm<=sm+1; end
    301: if (tb_ready) begin
            $display("DIST= %h, expecting 0",tb_data_out[15:0]);
            if(tb_data_out[15:0]!=16'h0000)error<=sm;
            sm<=sm+1; end
    302: if (tb_ready) begin
                  clusterID<=clusterID+1;
                  if (clusterID >(ncount/12)) clusterID<=16'hFFFF; // end of firing list
                  tb_read(NM_CAT);sm<=sm+1;
                  end
    303: if (tb_ready) begin
                $display("CAT= %d, expecting %d", tb_data_out[15:0], clusterID);
            if (tb_data_out[15:0]!=clusterID)
```

```
            begin
                 error<=sm;
                       sm<=sm+1;
                  end
                  else sm<=sm-3;
                  end
    304:    begin
            $display("%d : Results: Error %d",$time,error);
            $display("-------------------------------------");
            if (error !=0) G_error<=G_error+1;
            CS_n<=1'b1; sm<=sm+1; end
    305:    begin CS_n<=1'b0; sm<=SM_TESTEND; end

//---------------------------------------
// THE END....if G_Error=0!!!!!
//---------------------------------------
    400:    begin
            $display("TOTAL TEST ERRORS: ", G_error);
            $finish;
            end
endcase



//SUBSIDIARY FUNCTIONS USED BY ABOVE TEST ROUTINES


/*****************************************
Write function from the testbench cu
*****************************************/
task tb_write;
input [4:0]register_in;
input [15:0]data_in;
begin
@(negedge clock)begin CS_n<=1'b0;ds<=1;read<=0;register[3:0]<=register_in[3:0]; data[15:0]<=data_in[15:0];end
@(negedge clock)begin ds<=0;read<=1;end
while (~tb_ready) @(negedge clock)begin ds<=0;read<=1;end
@(negedge clock);CS_n<=1'b1;
end
endtask

/*****************************************
Read function from the testbench cu
*****************************************/
task tb_read;
```

```
input [3:0]register_in;
begin
@(negedge clock) begin CS_n<=1'b0;ds<=1;read<=1;register[3:0]<=register_in[3:0];end
@(negedge clock)ds<=0;
while (~tb_ready) @(negedge clock)ds<=0;
@(negedge clock);CS_n<=1'b1;
end
endtask


/*************************************************************
Broadcast a vector in normal mode
including its context and 4 components decoded from a 16-bit value
*************************************************************/
task PutVect4;
input [15:0]vectordata;
input [7:0]context;
      begin
    tb_write(NM_GCR,{8'h00,context[7:0]});
      tb_write(NM_COMP, {12'h000,vectordata[15:12]});
      tb_write(NM_COMP, {12'h000,vectordata[11:8]});
      tb_write(NM_COMP, {12'h000,vectordata[7:4]});
      tb_write(NM_LCOMP,{12'h000,vectordata[3:0]});
      end
endtask

/*************************************************************
Broadcast a vector in normal mode
including its context and veclen components decoded from a 16-bit value
*************************************************************/
task PutVectSlope;
input [7:0]offsetdata;
input [7:0]veclen;
input [15:0]learncat;
reg [7:0]j;
begin
    for (j=0;j<(veclen-1);j=j+1)
        begin
        @(negedge clock) tb_write(NM_COMP,{8'h00,offsetdata[7:0]}+{8'h00,j});
        end
    @(negedge clock) tb_write(NM_LCOMP,{8'h00,offsetdata[7:0]}+{8'h00,j});
    for (j=0;j<3;j=j+1) @(negedge clock) ds<=0;
    if (learncat[15:0]!=16'h0000) tb_write(NM_CAT,learncat[15:0]);
    for (j=0;j<19;j=j+1) @(negedge clock) ds<=0;
end
```

```
endtask

/*********************************************************
Broadcast a test vector in normal mode
featuring a sequence of 2 values repeated a number of times
*********************************************************/
task PutVectCst;
input [15:0]vectordata;
input [8:0]veclen;
reg [7:0]j;
    begin
    for(j=0;j<((veclen/2)-2);j=j+1)
        begin
        @(negedge clock); tb_write(NM_COMP,{8'h00,vectordata[7:0]}); //$display("SYSTEM PEAK at %d : vect comp#%d",
$time, j*2);
        @(negedge clock); tb_write(NM_COMP,{8'h00,vectordata[15:8]});  //$display("SYSTEM PEAK at %d : vect comp#%d",
$time, (j*2)+1);
        end
    @(negedge clock); tb_write(NM_COMP,{8'h00,vectordata[7:0]});//$display("SYSTEM PEAK at %d : vect comp#%d", $time,
254);
    @(negedge clock); tb_write(NM_LCOMP,{8'h00,vectordata[15:8]});//$display("SYSTEM PEAK at %d : vect comp#%d", $time,
255);
    end
endtask

/*****************************************
Write N components and category in SR mode
*****************************************/
task ramtestCM1K;
input [15:0]vectordata;
input [8:0]veclen;
input [15:0]category;
reg [7:0]j;
    begin
    @(negedge clock);tb_write(NM_NSR,16'h0010);
    for(j=0;j<(veclen/2);j=j+1)
        begin
          @(negedge clock); tb_write(NM_ALLCOMP,{8'h00,vectordata[7:0]});// $display("WORST TEST CASE at %d : RAM
comp#%d", $time, j*2);
          @(negedge clock); tb_write(NM_ALLCOMP,{8'h00,vectordata[15:8]});// $display("WORST TEST CASE at %d : RAM
comp#%d",$time,(j*2)+1);
        end
    @(negedge clock); tb_write(NM_ALLCAT,category[15:0]);
    @(negedge clock);tb_write(NM_RESETCHAIN,16'h0000);
    @(negedge clock);tb_write(NM_NSR,16'h0000);
```

```
    end
endtask

/***************************************
Write N components and category in SR mode
***************************************/
task ramtestNM500;
input [15:0]vectordata;
input [8:0]veclen;
input [15:0]category;

reg [7:0]j;
    begin
    @(negedge clock);tb_write(NM_NSR, 16'h0010);
    @(negedge clock); tb_write(NM_ALLCAT, category[15:0]); // category must be different from 0
    @(negedge clock);tb_write(NM_NSR, 16'h0000);
    for(j=0;j<(veclen/2);j=j+1)
        begin
            @(negedge clock); tb_write(NM_COMPINDEX,j*2);
              @(negedge clock); tb_write(NM_ALLCOMP,{8'h00, vectordata[7:0]});
            @(negedge clock); tb_write(NM_COMPINDEX,(j*2)+1);
            @(negedge clock); tb_write(NM_ALLCOMP,{8'h00, vectordata[15:8]});
        end
      // verify if next instruction is necessary (refer to Test_RAMandREG Arduino script)
      @(negedge clock);tb_write(NM_NSR, 16'h0000);
      end
endtask

/***************************************************************
Commit a 1st slope vector in SR mode
and fill the remaining neurons with a context 99 and dummy data
***************************************************************/
task fillclusterSR;
//assumes that the neurons are already in SR mode
input [7:0]ncr;
input [7:0]veclen;
input [15:0]cat;
reg [7:0]j;
begin
    @(negedge clock) tb_write(NM_NCR,{8'h00,ncr});
    for (j=0;j<veclen;j=j+1)
        begin
        @(negedge clock) tb_write(NM_COMP,j);
        end
    if (cat[15:0]!=16'h0000) tb_write(NM_CAT, cat[15:0]);
```

```
      // fill the remaining 11 neurons of the cluster
      for (j=0;j<11;j=j+1)
      begin
          @(negedge clock) tb_write(NM_NCR,99);
        @(negedge clock) tb_write(NM_COMP,1);
            @(negedge clock) tb_write(NM_CAT,1);
    end
end
endtask

/***********************************************
Write ctxt, 4 components, aif and cat in SR mode
***********************************************/
task writeSRneuron;
input [7:0]offset;
    begin
    tb_write(NM_NCR, {8'h00,offset[7:0]});
    tb_write(NM_COMP, {8'h00,offset[7:0]+1});
    tb_write(NM_COMP, {8'h00,offset[7:0]+2});
    tb_write(NM_COMP, {8'h00,offset[7:0]+3});
    tb_write(NM_AIF, {8'h00,offset[7:0]+4});
    tb_write(NM_CAT, {8'h00,offset[7:0]+5});
    end
endtask

/***************************************************
Write ctxt, aif, cat and veclen components in SR mode
Can be very long depending on veclen
***************************************************/
task writeSRneuron_vlen;
input [7:0]offset;
input [7:0]veclen;
reg [7:0]j;
    begin
    tb_write(NM_NCR, {8'h00,offset[7:0]});
    for (j=0; j<veclen; j=j+1)
    begin
        tb_write(NM_COMP, {8'h00,offset[7:0]+j});
    end
    tb_write(NM_AIF, {8'h00,offset[7:0]+4});
    tb_write(NM_CAT, {8'h00,offset[7:0]+5});
    end
endtask

endmodule
```