Project Name	**OF PN Project Name** **Project Name** **Block Name** **Digital High Level Design** **Version 0.1**

Amit Biran 305279093

Aviran Huga 302901517 **verification guide**

Version 0.1 4 June 2007

Revision Log

Rev	Change	Description	Reason for change	Done By	Date
0.1	Initial document			Shy Hamami	4,Jun,2007
0.2	Digital Changes			Amir Kolaman	14,Jul,2007

0.3	Functional Verification			Amir Kolaman	4,November 2007

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. VERIFICATION PLAN

In order to prove that our module works correctly and that it is stable and reliable we made a verification environment and tested the module in three main scenarios:

1)Standard Scenarios - 40 standard pictures are being loaded into the module.

2) Extreme Scenarios- non standard pictures are being loaded into the module we get non standard pictures by creating random pixel values.

3)Forbidden Scenarios- we test the module in a forbidden scenarios for example trying to load a picture while in reset mode.

Table 1- tests description

Test Number	Functionality being tested	Test data set	Expected result	Scenario
1	Read and Write proper values into the registers	Valid and random images	we want to bew able to read the correct data from each register in the address space	Standard Extreme
2	Reset is working correctly	any	CatRecOut not equal 1 while reset is active	Standard Extreme Forbidden
3	Valid calculation	Valid and random images, weight files	The value we calculate always equal to the sigma of the pixels times the weights.	Standard Extreme

1.1 Verification Test Objectives

The objective of our test is to determine whether the data is being loaded correctly into the module, if the output of the module is correct and if at any point of the calculation our module hold a correct value. Also we check that the board work as expected in reset mode means its ideal.,

1.2 Test Bench Architecture and Functionality

Our test has three main components:

1)Stimulus – communicate with the DUT feed data to it and change its states in our case it stimulate a CPU.

2)Checker – listens to the output of the DUT and make sure that its states are leagal and that the output is correct.

3)Coverager – make sure we cover all the functionality we need to in our DUT.

The way our testbench works is the following:

The stimulus first use reset for a while this give the checker a chance to make sure that the DUT is ideal after that, the stimulus loads data into the registers and triggers the start of the calculation by writing 1 into register in address 0. While the stimulus writes the data the checker makes sure that correct data was written into the registers (the checker has a reference to the registers), the way the checker does that is by finding that the PWRITE and PSEL are up then it knows that on the next clock the value on the PWDATA should be in the register of address PADDR and this is exactly what the checker check. While the calculation is running the checker preform a calculation on its own according to the data inside the registers(which was proven to be correct on the writing stage) each clock the checker makes sure that the value it calculated feats the data the module calculated (the checker has a reference to the accumulator which holds the value inside the module). Once finished the checker makes sure that the output feats the output it expects according to the calculation that happened inside the checker.

Between each picture the stimulus raise the reset which gives the checker another chance to check if the reset works correctly.

While the test is running the coverage makes sure we cover all the functionality. The manner in which this process happens will be discussed in the functional cover section of this report.

The way the stimulus checker DUT and coverage communicate is through the DUT interface means each module holds the same reference to the DUT input and output ports interface. This way the checker and coverage can know when stimulus loads data, raise reset and also when the DUT outputs new data.

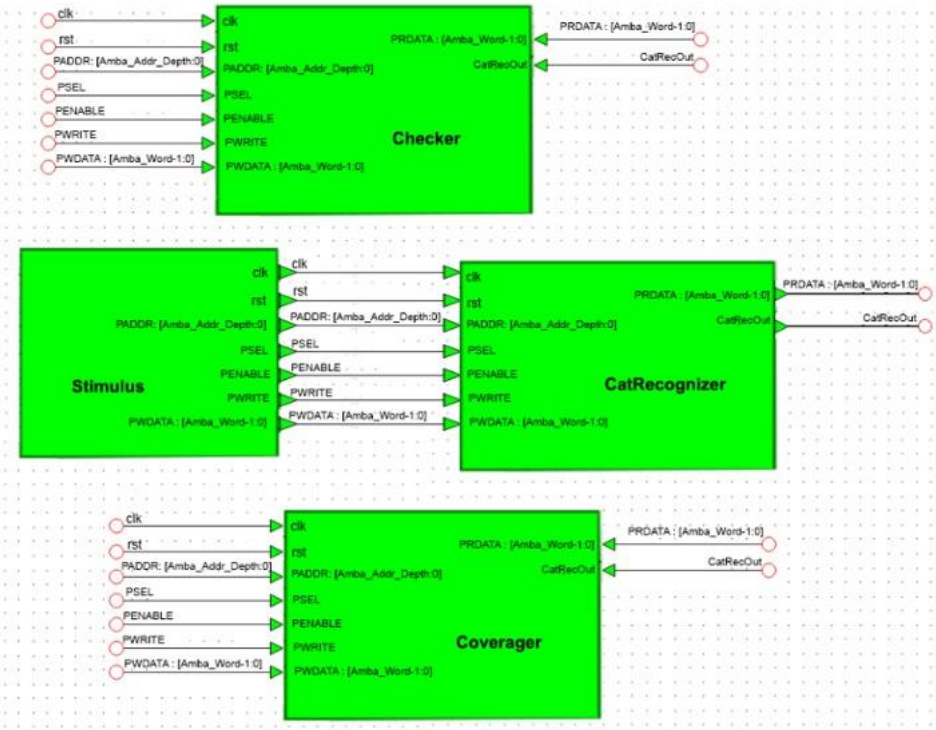

Figure 1-Test Bench Block Diagram

1.3 Functional Coverage

Given below is a description of our functional coverage:

Table 2- Test Plan Functional Coverage

FUNCTION	EVENT	COVERAGE POINT	BINS	scenario
rst_counter	**Posedge rst**	rst	0,1	**all**
PSEL_counter	**Posedge clock**	PSEL	0,1	**all**
PENABLE_counter	**Posedge clock**	PENABLE	0:1	**all**
address	**Posedge clock**	PADDR	[1500:0],[3000:1501],[4096:3001]	**Standard Extreme**

1.4 Test Bench Functional Checkers

Here, describe the functional Checkers for each of the test conditions we have covered, and also describe what are the expected results of the specific test case.

Condition – at what condition should the function be checked?

Expected – what are the expected results?

Table 3- functional checker

Condition	Event	Expected Result	Scenario
Reset Active	Posedge reset	@(checker_Interface.rst) (checker_Interface.rst==1) |-> (checker_Interface.CatRecOut!=1);	Standard
Valid registers value on write	Posedge clock	@(posedge checker_Interface.clk) checker_Interface.PSEL && !checker_Interface.PENABLE && checker_Interface.PWRITE && !first_time |-> (registers[last_Address]==last_value); (first time means is 1 until we find that calculation started then it changes to 0 until the next image is loaded)	Standard
Valid accumulator value	Posedge clock	@(posedge checker_Interface.clk) calc_counter>=1 && calc_counter <=4096 |-> calculated_acc_val == acc_val;	Standard
Valid output	Posedge clock	APB.CatRecOut == 1'b0 |-> (currentResult < 0)	Standard
Accumulate currentResult checker	Posedge clock	@(posedge checker_Interface.clk) checker_Interface.CatRecOut == 1 |-> (last_result>0);	Standard

2. VERIFICATION IMPLEMENTATION

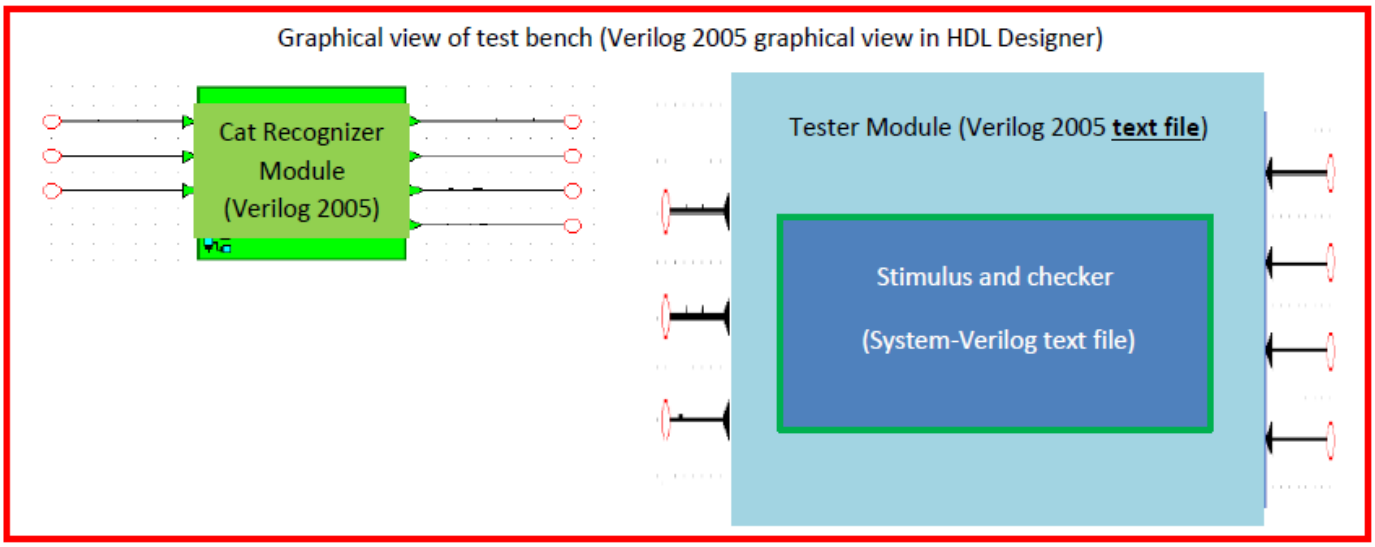

Figure 2- combining system verilog with graphical views of verilog 2005

2.1 Functional Coverage

Here is the code for the functional coverage:

```
`resetall
`timescale 1ns/10ps
module coverager#(
  parameter Amba_Word = 24,
  parameter Amba_Addr_Depth = 13,
  parameter Weight_precision = 5
)
(
  cat_recognizer_interface.checkr_coveragr coverager_Interface
);

covergroup cg @(posedge coverager_Interface.clk);
  rst_counter      : coverpoint coverager_Interface.rst
{
    bins high = {1};
    bins low = {0};
}

  PSEL_counter      : coverpoint coverager_Interface.PSEL
{
    bins high = {1};
    bins low = {0};
}

  PENABLE_counter      : coverpoint coverager_Interface.PENABLE
{
    bins high = {1};
    bins low = {0};
}

  PWRITE_counter      : coverpoint coverager_Interface.PWRITE
{
    bins high = {1};
    bins low = {0};
}

 address : coverpoint coverager_Interface.PADDR
 {
    bins low = {[1500:0]};
    bins med = {[3000:1501]};
    bins high = {[4096:3001]};

 }

endgroup


cg cov = new();
endmodule
```

Figure 3- functional coverage code

2.2 Stimulus

The code for the stimulus module is pretty long we decided to not add it to this document so it will stay organized and neat therefore please see stimulus_random.sv in the project. we already discussed the way in which we implemented the stimulus.

Below is a class we wrote to generate random numbers for the random images.

```
package classes;

  //*********************Begin Class*************************************
  // This is the class that we will randomize.
class RandomGenerator#( parameter size = 24);
   //parameter size = 24;

  rand  reg[size-1:0] value;          //regular random variable
//  rand inst_e inst;

  // Randomization constraints.
  /*
  constraint UpperLimit {
    dinClass<(1<<(range+1)-1);
    up_dnClass<2;
  }
  */

  // Print out the items.
  function void print();
    $display("value = %0h", value);
  endfunction

endclass
  //*********************End Class*************************************

endpackage
```

Figure 4- random generator class to randomize the data for the random input test

2.3 Interface

```
`resetall
`timescale 1ns/10ps
interface cat_recognizer_interface #( parameter Amba_Word = 24,
  parameter Amba_Addr_Depth = 13,
  parameter Weight_precision = 5
  )();

  logic PENABLE;
  logic PSEL;
  logic PWRITE;
  logic clk;
  logic rst;
  logic [Amba_Addr_Depth-1:0] PADDR;
  logic [Amba_Word-1:0] PWDATA;
  logic[Amba_Word-1:0] PRDATA;
  logic CatRecOut;

  //modports declaration
modport stimuls (input PENABLE,PSEL,PWRITE,clk,rst,PADDR,PWDATA);
modport cat (input PENABLE,PSEL,PWRITE,clk,rst,PADDR,PWDATA,
                output PRDATA,CatRecOut);
modport checkr_coveragr (input PENABLE,PSEL,PWRITE,clk,rst,PADDR,PWDATA,PRDATA,CatRecOut);



// ### Please start your Verilog code here ###
endinterface
```

Figure 5-APB interface

3. VERFICTION RESULTS

3.1 Functional coverage report and checkers coverage report :

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/tb_CatRecognizer/checker1/cover__valid_output	SVA	✓	Off	60	1	Unli...	1	100%		✓	0	0	0 ns	0
/tb_CatRecognizer/checker1/cover__valid_acc_val	SVA	✓	Off	61425	1	Unli...	1	100%		✓	0	0	0 ns	0
/tb_CatRecognizer/checker1/cover__valid_reg_value_write	SVA	✓	Off	69070	1	Unli...	1	100%		✓	0	0	0 ns	0
/tb_CatRecognizer/checker1/cover__reset_active	SVA	✓	Off	18	1	Unli...	1	100%		✓	0	0	0 ns	0

Figure 6- coverage

```
      COVERGROUP COVERAGE:
--------------------------------------------------------------------------------
------------
Covergroup                                                Metric       Goal     Status

--------------------------------------------------------------------------------
------------
 TYPE /tb_CatRecognizer/coverager1/cg                     100.0%        100     Covered
    covered/total bins:                                       11         11
    missing/total bins:                                        0         11
    % Hit:                                                100.0%        100
    Coverpoint cg::rst_counter                            100.0%        100     Covered
        covered/total bins:                                    2          2
        missing/total bins:                                    0          2
        % Hit:                                            100.0%        100
    Coverpoint cg::PSEL_counter                           100.0%        100     Covered
        covered/total bins:                                    2          2
        missing/total bins:                                    0          2
        % Hit:                                            100.0%        100
    Coverpoint cg::PENABLE_counter                        100.0%        100     Covered
        covered/total bins:                                    2          2
        missing/total bins:                                    0          2
        % Hit:                                            100.0%        100
    Coverpoint cg::PWRITE_counter                         100.0%        100     Covered
        covered/total bins:                                    2          2
        missing/total bins:                                    0          2
        % Hit:                                            100.0%        100
    Coverpoint cg::address                                100.0%        100     Covered
        covered/total bins:                                    3          3
        missing/total bins:                                    0          3
        % Hit:                                            100.0%        100
 Covergroup instance \/tb_CatRecognizer/coverager1/cov
                                                          100.0%        100     Covered
    covered/total bins:                                       11         11
    missing/total bins:                                        0         11
    % Hit:                                                100.0%        100
    Coverpoint rst_counter                                100.0%        100     Covered
        covered/total bins:                                    2          2
        missing/total bins:                                    0          2
        % Hit:                                            100.0%        100
```

```
        bin high                                          152          1    Covered
        bin low                                        199848          1    Covered
    Coverpoint PSEL_counter                            100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin high                                       138157          1    Covered
        bin low                                         61834          1    Covered
    Coverpoint PENABLE_counter                         100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin high                                        69070          1    Covered
        bin low                                        130921          1    Covered
    Coverpoint PWRITE_counter                          100.0%        100    Covered
        covered/total bins:                                 2          2
        missing/total bins:                                 0          2
        % Hit:                                         100.0%        100
        bin high                                       138157          1    Covered
        bin low                                         61834          1    Covered
    Coverpoint address                                 100.0%        100    Covered
        covered/total bins:                                 3          3
        missing/total bins:                                 0          3
        % Hit:                                         100.0%        100
        bin low                                        112909          1    Covered
        bin med                                         51000          1    Covered
        bin high                                        36080          1    Covered

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------
----------
Name                                      Design Design   Lang File(Line)      Count
Status
                                          Unit   UnitType
--------------------------------------------------------------------------------
----------
/tb_CatRecognizer/checker1/cover__valid_output
                                          Checker Verilog  SVA
C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_reco
gnizer_lib/hdl/checker.sv(73)
                                                                                 60
Covered
/tb_CatRecognizer/checker1/cover__valid_acc_val
                                          Checker Verilog  SVA
C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_reco
gnizer_lib/hdl/checker.sv(69)
                                                                              61425
Covered
/tb_CatRecognizer/checker1/cover__valid_reg_value_write
                                          Checker Verilog  SVA
C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_reco
gnizer_lib/hdl/checker.sv(65)
                                                                              69070
Covered
/tb_CatRecognizer/checker1/cover__reset_active
```

```
                                        Checker Verilog  SVA
C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_reco
gnizer_lib/hdl/checker.sv(61)
                                                                                 18
Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 4
```

We got 100 percent of functional coverage and the testbench runs for 200us without errors which means that our design is on the right direction but obviously we cant be sure this verification has taken a few days while in real companies they preform verification for months.

```
# //  INFORMATION THAT ARE PRIVILEGED, CONFIDENTIAL, AND EXEMPT FROM
# //  DISCLOSURE UNDER THE FREEDOM OF INFORMATION ACT, 5 U.S.C. SECTION 552.
# //  FURTHERMORE, THIS INFORMATION IS PROHIBITED FROM DISCLOSURE UNDER
# //  THE TRADE SECRETS ACT, 18 U.S.C. SECTION 1905.
# //
# vsim -f hds_args.tmp -foreign "hdsInit C:/MentorGraphics/HDS_2012.1/resources/downstream/modelsim/ModelSim_64Bit.dll" -pli ""C:/MentorGraphics/HDS_2012.1/resources/downstream/modelsim/ModelSim_64Bit.dll""
# Start time: 16:21:37 on Dec 21,2018
# ** Warning: (vsim-8891) All optimizations are turned off because the -novopt switch is in effect. This will cause your simulation to run very slowly. If you are using this switch to preserve visibility for Debug or PLI features please see the User's Manual section on
Preserving Object Visibility with vopt.
#
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.tb_CatRecognizer
# Loading sv_std.std
# Loading cat_recognizer_lib.tb_CatRecognizer
# Loading cat_recognizer_lib.cat_recognizer_interface
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.cat_recognizer_wrapper
# Loading cat_recognizer_lib.cat_recognizer_wrapper
# Loading cat_recognizer_lib.cat_recognizer
# Loading cat_recognizer_lib.fsm_apb
# Loading cat_recognizer_lib.RegisterFile
# Loading cat_recognizer_lib.weights_memory
# Loading cat_recognizer_lib.NeuronCalculator
# Loading cat_recognizer_lib.Neuron
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.stimulus_random
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.stimulus_random_sv_unit
# Loading cat_recognizer_lib.classes
# Loading cat_recognizer_lib.stimulus_random_sv_unit
# Loading cat_recognizer_lib.stimulus_random
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.Checker
# Loading cat_recognizer_lib.Checker
# Refreshing C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/work.coverager
# Loading cat_recognizer_lib.coverager
# ** Warning: (vsim-3015) C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/hdl/cat_recognizer.v(79): [PCDPC] - Port size (13) does not match connection size (32) for port 'address'. The port definition is at: C:/Users/amitb/
Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/hdl/weights_memory.v(13).
#    Time: 0 ns  Iteration: 0  Instance: /tb_CatRecognizer/behavioral_cat_recognizer_o/behavioral_cat_recognizer/weight_mem File: C:/Users/amitb/Desktop/UNIV/year4/logi/temop/Cat_Recognizer/cat_recognizer/cat_recognizer_lib/hdl/weights_memory.v
# Loading C:/MentorGraphics/HDS_2012.1/resources/downstream/modelsim/ModelSim_64Bit.dll
VSIM 1> run
# value = 38c0c4
# value = 1c81e3
# 0
#
coverage report -detail -cvg -directive -comments -file fcover_report.txt -r /
```

Figure 7- verification result

4. GOLDEN MODEL

Result of our golden model vs our results:

```
VSIM 3> run
# value = 38c0c4
# value = 1c81e3
# acc =                    x,cat_out = x
# acc =                -4048,cat_out = 0
# acc =                 9127,cat_out = 1
# acc =                12207,cat_out = 1
# acc =                 4130,cat_out = 1
# acc =                13085,cat_out = 1
# acc =                 4412,cat_out = 1
# acc =                 3486,cat_out = 1
# acc =               -15694,cat_out = 0
# acc =                 5967,cat_out = 1
# acc =                 7126,cat_out = 1
# acc =                 3017,cat_out = 1
# acc =                -6603,cat_out = 0
# acc =                 4136,cat_out = 1
# acc =                 3792,cat_out = 1
# acc =                13159,cat_out = 1
# acc =               -20274,cat_out = 0
# acc =                13930,cat_out = 1
# acc =               -19446,cat_out = 0
# acc =                 6504,cat_out = 1
# acc =                -7169,cat_out = 0
# acc =               -14244,cat_out = 0
# acc =                 7399,cat_out = 1
# acc =                -8543,cat_out = 0
# acc =               -23025,cat_out = 0
# acc =                 8518,cat_out = 1
# acc =                -1111,cat_out = 0
# acc =                -5527,cat_out = 0
# acc =                 5640,cat_out = 1
# acc =               -22590,cat_out = 0
# acc =               -10466,cat_out = 0
# acc =                29638,cat_out = 1
# acc =               -11651,cat_out = 0
# acc =                 6613,cat_out = 1
# acc =                  120,cat_out = 1
# acc =                 -182,cat_out = 0
# acc =                 7594,cat_out = 1
# acc =               -17325,cat_out = 0
# acc =               -14217,cat_out = 0
# acc =                 4987,cat_out = 1
# acc =                -7494,cat_out = 0

VSIM 4>
```

```
C:\Users\amitb\AppData\Local\Programs\Python\Python36\python.exe C:/User
9127
12207
4130
13085
4412
3486
-15694
5967
7126
3017
-6603
4136
3792
13159
-20274
13930
-19446
6504
-7169
-14244
7399
-8543
-23025
8518
-1111
-5527
5640
-22590
-10466
29638
-11651
6613
120
-182
7594
-17325
-14217
4987
-7494
-17535

Process finished with exit code 0
```

Figure 8- golden model vs verification result

The first test in the verification is a random image. This explain the first result that doesn't match anything in golden model. It is possible to see that the rest of the results match our golden model and outputs the correct value in CATRECOUT.

Code of our golden model:

```python


file_to_read = "WeightsPrecision5.txt"
file_pixels = "Image0.txt"


for im_num in range(0,40):
    file_pixels = "Image{}.txt".format(str(im_num))

    with open(file_to_read,'r') as f:
        for line in f:
            weights_values = f.readlines()

    for i in range(len(weights_values)):
        weights_values[i] = int(weights_values[i])


    with open(file_pixels,'r') as f:
        for line in f:
            pixels_values = f.readlines()

    for i in range(len(pixels_values)):
        pixels_values[i] = int(pixels_values[i])


   # print (weights_values)
   # print (pixels_values)

    #print(len(weights_values),len(pixels_values))

    acc = 0
    for i in range(len(pixels_values)):
        acc += pixels_values[i]*weights_values[i]
        #if(i%3 == 2):
        #print(int(i/3),acc,pixels_values[i],weights_values[i],pixels_values[i]*weights_values[i])
    print(acc)
```

Figure 9- code for golden model

5. APPENDIX

5.1 Terminology

LSB - Least Significant Bit

TBR - To Be Reviewed

TBD - To Be Defined

5.2 References