UVM for Kmeans IP

[Document subtitle]

[Date]

Liora Huf

Edi Sraiber

Led by: Goel Samuel

Contents

[1 Hardware verification 1](#_Toc52869232)

[1.1 What,why,how? 1](#_Toc52869233)

[2 UVM 2](#_Toc52869234)

[2.1 Structure 2](#_Toc52869235)

[2.2 Coverage 2](#_Toc52869236)

[3 DUT 3](#_Toc52869237)

[3.1 Architectural description 3](#_Toc52869238)

[3.2 Details of parameters that can be varied 3](#_Toc52869239)

[3.3 Description of APB protocol 3](#_Toc52869240)

[4 Implemented Verification environment 4](#_Toc52869241)

[4.1 UVM used classes 4](#_Toc52869242)

[4.2 Ref Model 4](#_Toc52869243)

[5 Test Plan 5](#_Toc52869244)

[5.1 Test Line 1 – Basic Test 5](#_Toc52869245)

[5.2 Test Line 2 – Out as In Test 5](#_Toc52869246)

[5.3 Test Line 2 – Robustness Test 5](#_Toc52869247)

[5.4 6](#_Toc52869248)

[6 Results 7](#_Toc52869249)

[7 Bug Fixes 8](#_Toc52869250)

[8 Conclusion 9](#_Toc52869251)

[9 Bibliography 10](#_Toc52869252)

[10 Appendix A – How to integrate Matlab code to UVM enviroment 11](#_Toc52869253)

# Hardware verification

TBD

## What,why,how?

TBD

# UVM

TBD

## Structure

TBD

## Coverage

TBD

# DUT

TBD

## Architectural description

TBD

## Details of parameters that can be varied

TBD

## Description of APB protocol

TBD

# Implemented Verification environment

TBD

## UVM used classes

TBD

## Ref Model

TBD

# Test Plan

Description of all scenarios to be defined : which parameters will be exercised.

         How is equivalence defined

         Description of Assertions to test APB interface

## Test Line 1 – Basic Test

In this test line, the following parameters will be randomly generated:

1. Number of points
2. Points values
3. Initial Centroid values

These randomly generated parameters should be sent to the DUT and the REF Model. The outputs given by the DUT and the REF Model for the mentioned input shall them be compared, being equivalent if every output centroid presented by the DUT is also presented by the REF Model.

This test line shall produce overall ten tests where each test the randomly generated parameters shall have different values.

## Test Line 2 – Out as In Test

In this test line, the following parameters will be randomly generated:

1. Number of points
2. Points values
3. Initial Centroid values

These randomly generated parameters should be sent to the DUT and the REF Model. After verifying that the outputs given by the DUT and the REF Model for the mentioned input shall are equivalent (every output centroid presented by the DUT is also presented by the REF Model), use the received outputs as input for a new run.

The results expected for the new run are:

1. The output centroids are equal to the input centroids
2. Convergence reached within one iteration

This test line shall produce overall ten tests where each test the randomly generated parameters shall have different values.

## Test Line 2 – Robustness Test

In this test line, the following parameters will be randomly generated:

1. Number of points
2. Points values
3. Initial Centroid values

These randomly generated parameters should be sent to the DUT and the REF Model. The outputs given by the DUT and the REF Model for the mentioned input shall them be compared, being equivalent if every output centroid presented by the DUT is also presented by the REF Model.

This test line will produce multiple tests which will be run in series without breaks.

## 

1. Test Lines
   1. Basic - randomly generate:

Insert these 3 parameters to DUT and REF Model, check for differences

Run this test 10 times?

* 1. Back to back - randomly generate:
     1. Number of points
     2. Points values
     3. Initial Centroid values

Get the result and do another iteration using it as initial centroids (with the same generated points)

Run this test 10 times?

* 1. Robustness - randomly generate:
     1. Number of points
     2. Points values
     3. Initial Centroid values

And run the algorithm multiple times in a row.

* 1. One iteration run - randomly

# Results

TBD

 Results of tests/scenarios run – bug identification

         Results of coverage

         Results of assertions

# Bug Fixes

TBD

1. Bug Fixes:

During the verification process, the following bugs have risen and handled:

* 1. Fix sign representation of variables:

During the caculation, each data point vector to 7 cordinates which shall be represented in fixed point and signed (See reference to chapter blab la in DUT chapters).

* + 1. The variable type of those coordinates were represented in unsigned(default of type in system Verilog is unsigned unless stating "signed" in the type, i.e. signed + type.
    2. The reason for the bug was since it was belived that the compiler will fit to 2's complement when arithmetic operations are being done, yet it did not happened and after diving in a debug process it came up.
    3. The solution was simple in this case and a "signed" syntax was added accordingly for each parsed coordinate process, it shall be noted that as a concatenated vector, the sign does not hold meaning since it matters in coordinate resolution.
    4. The file "accumulator\_adder.sv" changed, as explained above.
  1. Fix 2's complement representation of numbers:
     1. In the summation process of points to form the nominator of the next developed centroids for each iteration, each coordinate holds 22 bits per coordinate(21 + 1 for sign), when each point hold 13(12 + 1 for sign).

See reference to chapter blab la in DUT chapters.

* + 1. When performing arithmetic operations to sum, a negative number represented in 2's complent with 13 bits, wasn’t handled to fit for the operation to be summed to 22 bits number.
    2. The fix was to handly transform the number to its absolute value, then creating the same value in 2's complement representation in 22 bits, then perform arithmetic operations to sum.
    3. The file "distance\_calc.sv" changed, as explained above.

# Conclusion

TBD

# Bibliography

TBD

# Appendix A – How to integrate Matlab code to UVM enviroment