1. Hardware verification
   1. What,why,how?
2. UVM
   1. Structure
   2. Coverages
3. DUT)brief explanation about it)
4. Implemented Verification environment
   1. UVM used classes
   2. Ref Model
5. Test Lines
   1. Basic - randomly generate:
      1. Number of points
      2. Points values
      3. Initial Centroid values

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 generate eight data points which will be used also as the eight initial centroids, multiple times in a row. Verify that in all runs convergence is reached in one iteration and final centroids are equal to initial centroids.
  2. Threshold - randomly generate:
     1. Number of points
     2. Points values
     3. Initial Centroid values
     4. Threshold value

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

* 1. Isolated centroid(“K value change”) - randomly generate:
     1. Number of points
     2. Points values
     3. Initial Centroid values
        1. Where one of the centroids is constrained to be far away from the all the data points. Verify its values does not change (no points are assigned to it)
        2. Where all of the centroids (except from one) are constrained to be far away from the all the data points. Verify their values does not change (no points are assigned to it)

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.

1. Conclusions
2. Bibliography