Project for Hardware-Software Codesign

Description: Create a Hardware-Software Codesign version of the k-mean clustering algorithm

K-means clustering is a popular data mining algorithm that partitions *n* samples into *k* clusters (note: the k-nearest neighbor classifier algorithm used in machine learning can leverage the cluster centers produced by the k-means clustering algorithm). The problem is in general NP-hard but heuristic algorithms have been developed that quickly converge to a local optimum solution. We will consider one of those algorithms in this project.

I have provided a C code version of the k-means clustering algorithm that you will use as a starting point. You must also report the performance of this version WITHOUT modifications in your report, using the 'gettimeofday' routine that I discussed in HISTO lab0.

You will need to study the C version and then decide which components to implement as a VHDL module using the BRAM memory that we created in HISTO lab0. The lower 4096 16-bit memory locations are available for use in the existing configuration (from HISTO lab0) but you should feel free to expand the size as needed. Details on how to reconfigure the memory size have been discussed in the lab0 video projects.

I have provided a link to wikipedia for your reference, and a paper published in 2011 that describes techniques to map machine learning algorithms, including k-means clustering, probability distribution functions and correlation algorithms, into hardware-software designs, optimized using advanced techniques such as pipelining. I think the paper is a very good reference but I recommend that you do not attempt to do any of the advanced techniques described UNTIL you have a working version of your algorithm. Extra credit will be given to any advanced techniques beyond those described in this project description, including the addition of any other type of IP that you find useful, such as FIFOs, additional GPIO registers, etc.

I have also provided at least one data set along with the initial assignment of cluster centroids that YOU MUST use as the initial test case(s). You are free to add additional test cases to demonstrate any features of your implementation but you must first demonstrate correct operation on the supplied test cases.

You may create teams of size at most 2 for this project (you and one other partner, or you may wish to work alone). If you choose to do so, you need to notify me as soon as possible (no later than the friday before the start of Module 8). Students who choose to work by themselves will be given a handicap regarding grading, i.e., I will expect approximately half as much from groups of size 1. All groups or individuals must provide a working version of the algorithm, with at least half of the algorithm running in the PL side.

Although you must write the VHDL code by hand (you cannot use HLS for this component), you are allowed (and encouraged) to use HLS for a comparative analysis in which you compare the performance and resources of your implementation with that predicted by HLS. Any group or individual who includes a custom vs. synthesized analysis in their report **will receive 10 extra credit points**.

Report Requirements:

1) You must turn in a written report as a PDF, which includes the following components:

- A description of the algorithm and the details of your implementation, i.e., which components did you implement in VHDL, how did you partition the BRAM memory, what types of performance analysis techniques did you use to identify components that are best moved to the PL side, etc?
- A section describing your results, including graphs. This section must show results from the test case(s), and then you may include additional results.
- A performance analysis section that gives runtimes (as we did in HISTO lab0) of BOTH the software ONLY version (you MUST use the C version that I have supplied WITHOUT changes) AND the hardware-software version that you developed (and potentially the HLS version -- see above for extra credit). You should attempt to separate data transfer times across the GPIO interface from actual hardware runtimes, as we did in the HISTO lab0 demo, when possible. **The team or individual with the most significant speedups over the software version will receive 10 extra credit points.**
- Optionally provide a video that shows a live demonstration of your project, as I have done in HISTO lab0. This will help ensure that I do not miss any additional features that you may have added. Cell phone videos are fine but please ensure they are decent quality.

I will provide updates to this project description as needed so please download periodically to check for updates. I will date and highlight any updates in red.