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, and a Vivado block diagram and memory layout (explained below) that you will use as a starting point. You will need to study the C version and then decide which components to implement as a VHDL module using the BRAM (you also used BRAM in HISTO lab0).

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 work with a partner, 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. HLS 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, any additions you have made to the BRAM, what types of performance analysis techniques did you use to identify components that are best moved to the PL side, etc?

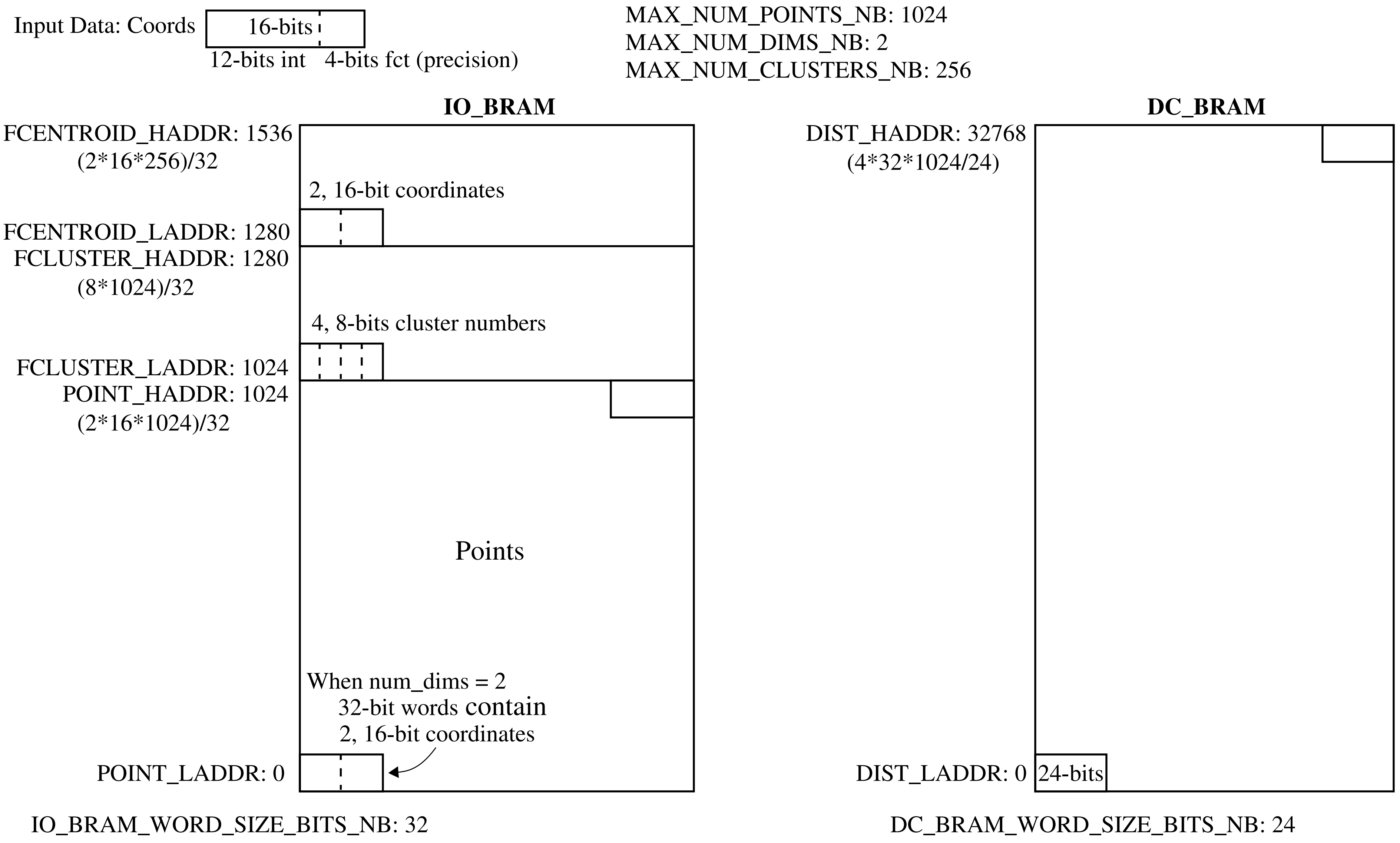
- A section describing your results, including graphs. This section must show results from the test case(s), and then you may include additional results.
- A performance analysis section that gives runtimes (as we did in HISTO lab0) of BOTH the software ONLY version (you MUST use the C version that I have supplied WITHOUT changes) AND the hardware-software version that you developed (and potentially a comparison with an 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.

See next page for BRAM memory layout.

BRAM MAX SIZES:
ZYBO: 270 KB or 2.1 Mbits
CORA: 225 KB 1.8 Mbits

BRAM:
The BRAM memory regions, with constants in DataTypes_pkg.vhd that partition the BRAMs as shown in the diagram below:



With MAX_NUM_DIMS_NB set to 2, each point is defined by 2 coord(inates). If you want to increase the number of dimensions, do so with even numbers, e.g., 4, 6, 8, etc.

NOTE: Only 5-bits of the 8-bit cluster numbers are used, so the actual MAXIMUM number of clusters is 32, not 256

Observations and Recommendations:

- I want everyone to use the current BRAM layout to do this project at the very minimum. You can expand on it if you like, but you must write the state machines that use the BRAM regions I've provided to compute intermediates, e.g., distances and current cluster assignments. Previous classes did not have this starter code. This should make it much easier for you to complete the project by the deadline.
- The kmean_codesign_starter.c code reads a data file, transfers it to the points region of IO BRAM, and then reads it back and prints it. I have not tested the DC BRAM LoadUnloadBRAM but it is nearly identical to the IO BRAM. Let me know if you see any problems.
- I've included a Makefile in the C_code directory (for 2019.1). You'll need to set the variables at the top to point your version of the compiler. Once done, just type 'make'. There is a README.jim file in there too for common commands to generate and transfer the bitstreams and binary to the board.
- Use the LoadUnloadBRAM functions as needed to debug your VHDL.
- Only 39% of the BRAM is used in my starter project -- feel free to increase the size as needed. Avoid using ALL available BRAM. It will make synthesis very slow or even fail.
- You may want to add another 256 32-bit words for the 'prev' cluster assignments to IO BRAM so you don't need to copy them back to the C program during the computation.
- I've included an MMCM in the block diagram and set it to 50 MHz. Feel free to increase it - you can double click and increase the output clock frequency if you wish, until you fail timing!
- I've include a VHDL version and a Verilog version of the starter HDL code and Vivado project. You only need to use one of these for the project. Most of you will use the VHDL version but feel free to use the Verilog version if you feel more comfortable with Verilog.
- If I have time, I'll post a DMA version that'll speed up the transfer between DDR and the BRAM.

Keep an eye out for updates to the project. I'll post an announcement if/when something changes.

Good luck