LU Decomposition on UMS – Programming Experience

1. One PE, largest square matrix to fit in local memory [due 11/9]
   1.1 Compare performance with same matrix in DRAM
2. Multiple PEs (within a Quad), matrix in DRAM [due 11/9]
   2.1 Use block decomposition
      - Use PE to do data transfer from DRAM
      - Use MTE to do data transfer from DRAM
      - What is the difference in performance? Is it dependent on the size of the matrix? Show at least three cases.

   2.2 Use cyclic decomposition
      - Use MTE to do data transfer

   2.3 Report differences in performance between block and cyclic decomposition using MTEs – execution cycles, load balance, utilization of CPU, bus and I/O utilization, bottlenecks, scalability, etc.
3. One PE and DSE, matrix in DRAM [due 11/23]
4. Multiple PEs (across Quads), matrix in DRAM [due 11/23]
5. Can you use double buffering effectively here? Implement using double buffering by using MTE? [due 11/23]
6. Use self scheduling for multiple PEs. What is a good size for the number of rows in block? [due 12/7]
7. Use PEs, DSEs, and MTEs to get the best performing LU. Show scalability [due 12/7]

Please follow the following:

1. Well documented like the tutorial examples.
2. You can copy from the tutorials. Use the same terminology and conventions.
3. When doing the performance runs, do not include time for initialize (??) and check (this is easy).
4. Use matA as the name of the matrix. The name of the “check” file should be lu.dat
5. We will use different initialize and check routines for validation
6. For scalability use at least three data points
7. All programs will be automatically checked for cheating using state of the art tools.
8. You might want to start with a C-program using other tools like Visual C or gcc along with their debuggers. Once that works, port to UMS.
9. Always verify your program with multiple input/outputs. We will provide a couple on the web.
10. Use small matrix size for verification since the simulation time for large matrices will be large.