First results from Nallatech H101-PCIXM FPGA accelerator

Anton Shterenlikht

Mech Eng Dept, Bristol University, University Walk Bristol BS8 1TR
Tel: 0117 928 8233   Email: mexas@bris.ac.uk

Background and Introduction

The author has developed the so-called Cellular Automata Finite Element (CAFE) model for transitional ductile to brittle fracture in steel [1,2]. The model utilises the finite element (FE) method to simulate structural strain gradients and stress fields using the local approach to fracture. The brittle fracture propagation in the CAFE model is simulated explicitly with Cellular Automata (CA).

The FE part of the CAFE model requires 64-bit floating point precision due to the presence of very large (mean stress) and very small (porosity change) quantities in some local approach expressions. The FE computation involves iterative solution of a system of non-linear PDEs. Therefore the FE part of the CAFE model is computationally expensive with significant demands on memory.

In contrast the CA part of the CAFE model uses byte integers to store grain orientation or class. However, many CA cells are required to adequately represent the microstructural details. For example, 1 cm$^3$ of material can require in the order of 10$^9$ CA cells. Accordingly the CA part of the CAFE model is computationally very cheap, but with very high demands on memory.

The author has acquired a Nallatech H101-PCIXM FPGA accelerator aiming to speed-up his CAFE code. This presentation covers initial assessment of this board according to the 3 criteria outlined above:
(a) acceleration of 8-bit integer operations, (b) acceleration of 64-bit floating point operations, and (c) overhead associated with splitting very large arrays into chunks which can fit into the board's memory.

Methods of assessment

The acceleration of 8-bit integer operations is assessed on the author's microstructure generation code. This program simulates 3D grain growth (solidification) following very simple CA rules. The FPGA's performance is compared with that of the fortran90 code on the host machine for several 3D array sizes.

The acceleration of 64-bit floating point operations is assessed using matrix multiplication. The FPGA's performance is compared with that of the fortran90 code with intrinsic function MATMUL and using DGEMM BLAS routine for several matrix sizes.

The overhead associated with processing large arrays is assessed by running the above codes with arrays greater that the board's memory.
The application development for H101-PCIXM FPGA included several stages. Firstly, since the author's codes are written in fortran90, and board at present supports only C, the f2c converter was used to obtain the C code. The second stage involved C to FPGA compilation using Nallatech's DIME-C compiler. In the third step the executables were generated with Nallatech's DIMEtalk Application Builder. Finally, the executables were run from a linux host machine.

The presentation will include major technical details of the H101-PCIXM FPGA board, illustration of the application development for the microstructure generation code, timings and discussion of results.

Acknowledgments

The author gratefully acknowledges financial support from The Royal Society's Research Grant.
