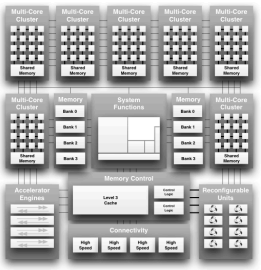
Mapping Simulation Algorithms to NoC MPSoC Computers

Prof. Dr. Hans-Joachim Wunderlich
Dipl.-Inform. Claus Braun

Motivation

Semiconductor technology scaling forces the development of highly parallel computer architectures. The explicit parallelism of these architectures is the key to continuously increasing computational performance, within a reasonable power envelope. These new parallel (NoC) MPSoC architectures provide a remarkably high computational performance and are particularly interesting for simulation applications.



Challenges:

- Nano-scaled semiconductors are increasingly prone to variability, vulnerability and reliability issues
- Fault tolerance mechanisms are mandatory from the hardware up to the software and the algorithm

Goals:

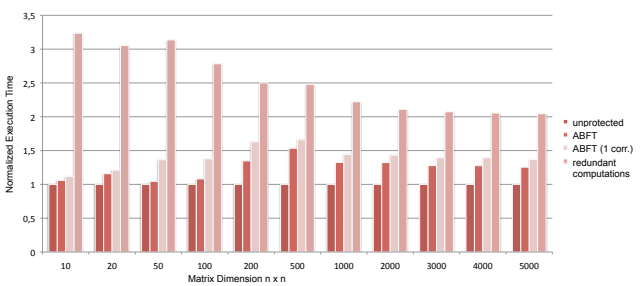
- Bring complex simulation applications to the desktop and mobile devices
- Mapping of fault tolerant core algorithms to NoC MPSoC architectures

State of Current Work

Mappings of fault tolerant core algorithms from linear algebra have been successfully developed for MP architectures like General-Purpose Graphics Processors (GPGPU) to cope with the reliability issues [1] and [2].

Algorithm-based Fault Tolerance (ABFT)

- Example: Block-based matrix operations, where data is encoded by checksums [3]
- Mappings of such fault tolerant algorithms were developed and are provided as C/CUDA program library (*speedway*)
- Overhead comparison between redundant computations and ABFT protected matrix multiplication:

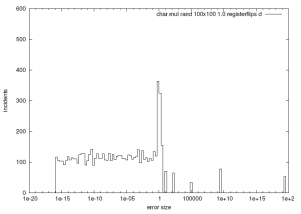


Optimization of Fault Detection Metrics

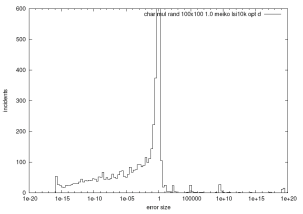
- Floating-point computations are prone to rounding errors
- Consequence: checksums of correct results can differ from reference checksums, leading to increased rate of false-positives and higher overhead
- Classical estimations of rounding error bounds are often too weak to achieve high error detection rates
- Solution: determination of appropriate metrics for error detection, e.g. the relative error between checksums in combination with thresholds
- Goal: optimal metrics and thresholds

Hardware Characterization through Simulation

Gate-level logic simulations are used in combination with fault injection campaigns to investigate the impact of single event transients on hardware like floating-point processing units (FPU).



simulation of register bit flips



simulation of gate-level bit flips

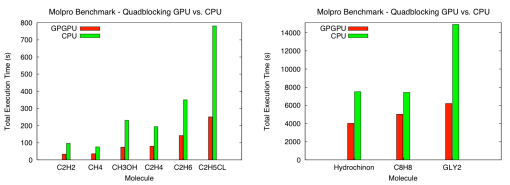
Future Work / Outlook

To provide overall fault tolerant algorithms on NoC MPSoC architectures, the focus of our research will be extended to the protection of the control flow. Here, we will concentrate on the development of software-based monitoring.

Cooperation

Institute for Theoretical Chemistry (Group of Prof. Werner)

- Acceleration of matrix operations in the Molpro quantum chemistry package, using General-Purpose Graphics Processors (GPGPU) [completed]



- Parallelization of Molpro module for density functional theory (DFT) calculations on GPGPU-based computer systems [curr. active cooperation]

ITI Computer Engineering Dept. (Prof. Radetzki, Adan Kohler)

- Mapping of fault tolerant simulation algorithms to NoC-based MP architectures, using MPI primitives [planned]

Connection to SimTech Visions

The remarkable computational power of parallel MPSoC computer architectures has the potential to drive all disciplines within the SimTech cluster and to open new fields of application for simulation technology outside of traditional computing centers. Our project builds the required bridges between these architectures and complex simulation applications to allow the reliable use of this potential.

Literature

[1] Claus Braun and Hans-Joachim Wunderlich, "Algorithm-based fault tolerance for many-core architectures," in 15th IEEE European Test Symposium (ETS), Prague, 2010, p. 253.

[2] Claus Braun and Hans-Joachim Wunderlich, "Algorithmen-basierte Fehlertoleranz für Many-Core-Architekturen," it - Information Technology, vol. 52, no. 4, pp. 209-215, August 2010.

[3] K. H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," IEEE Transactions on Computers, vol. C-33, no. 6, pp. 518-528, June 1984.