Making bug-checking in software and hardware design cheaper and more efficient

March 15 2022

Credit: CC0 Public Domain

The development of complex hardware and software is error-prone and costly. Testing can detect the presence of bugs in these designs, but it cannot prove their absence. One technique that can provide worthful feedback on the correctness of system designs is model checking. Model checking is an automated reasoning technique to find flaws in hardware and software systems. Ph.D. candidate Muhammad Mahmoud has redesigned algorithms to make them more suitable for model checking using GPUs, which allow for parallel computing at low cost.

Model checking is used to catch potential bugs as early as possible—preferably at the design phase—to make the necessary modifications quickly and cost-effectively. Successful examples of model checking include verifying CERN controllers, railway interlockings, nuclear control systems, and medical imaging. Companies such as Amazon, Microsoft, and Facebook use and develop model checking technology to ensure their products behave functionally correct.

Bounded model checking

However, model checking is computationally very demanding. It involves exhaustively analyzing a system design to determine whether it satisfies desirable functional specifications.

Bounded model checking (BMC) is a contemporary symbolic technique that can analyze large designs in reasonable time. BMC determines whether a model satisfies a certain property expressed in temporal logic, by translating the model checking problem to a propositional satisfiability (SAT) problem, for instance.

In this thesis, Muhammad Mahmoud, of the research group Software Engineering and Technology at the department of Mathematics and Computer Science, investigated how Graphics Processing Units (GPUs) can be employed effectively for BMC, focusing on the reasoning on SAT. GPUs offer great potential for parallel computation, while keeping power consumption low.

However, not all types of computation can trivially be performed on GPUs, in most applications the algorithms need to be entirely redesigned.

Simplifying the formulas

The researcher focused on the simplifications of SAT formulas, a strategy that leads to a drastic reduction of the formula size and the search space.

Next, he presented a new SAT solver which rigorously interleaves the search with so-called inprocessing. Inprocessing has proven to be powerful in modern SAT solvers, particularly when applied on SAT formulas encoding software and hardware verification problems.

The new solver is hybrid, capable of running the parallel part on the GPU while the actual solving will run sequentially on the Central Processing Unit (CPU).

In his thesis, Mahmoud also discusses the design aspects of the data structures and the memory management of our parallel implementations, leading to substantial improvements in execution performance.

Multiple decision making

Concerning the solving part, he extended the Conflict-Driven Clause Learning (CDCL) search algorithm with multiple decision making (MDM).

The MDM procedure has the ability to make and propagate multiple decisions at once. Moreover, it is augmented with local search to improve the accuracy in assigning truth values to these decisions.

Finally, he integrated the solver to a state-of-the-art bounded model checker. After optimizing further the inprocessing engine and making the solving process incremental, he investigated the impact of GPU-enabled BMC on software verification using Amazon Web Services (AWS) C99 library.

More information: GPU Enabled Automated Reasoning. research.tue.nl/files/19516561 … 0310_Muhammad_hf.pdf

Provided by Eindhoven University of Technology