

# I compute, therefore I am (buggy): methodic doubt meets multiprocessors

Jade Alglave<sup>1,2</sup>   Luc Maranget<sup>3</sup>   Daniel Poetzl<sup>4</sup>   Tyler Sorensen<sup>2</sup>  
<sup>1</sup> Microsoft Research Cambridge   <sup>2</sup> University College London  
<sup>3</sup> INRIA   <sup>4</sup> University of Oxford

## ABSTRACT

As a community, we (programmers, compiler writers, hardware architects, etc.) often take folklore, e.g. claims in programming guides, for granted. Inspired by Descartes' *methodic doubt*, i.e. challenging the truth of one's beliefs, we question this folklore. Thus, we have developed a tool suite to systematically test the memory ordering behaviour of multi- and manycore chips and compared our observations to what appears in authoritative documents.

To illustrate our approach, we passed the current paragraph to a program which concurrently ciphers, then deciphers, a piece of text on a graphics processing unit (GPU). It uses a mutex, i.e. mutual exclusion mechanism, given in the popular educational book *CUDA by Example* [3]. It is easy to see that some of the ciphered text remains; this is due to a bug in the published mutex which allows threads to read stale values in critical sections. We discovered this buggy behaviour (amongst others) during a large empirical study of deployed GPUs [1]. While our example is for GPUs, we first developed the approach for CPUs, notably IBM Power and ARM chips [2].

We then sent the present paragraph through the same cipher program; this time we fixed the bug by adding synchronisation instructions to the mutex (programs available at <http://www0.cs.ucl.ac.uk/staff/T.Sorensen/TinyToCS3>); no ciphered text remains. Indeed, our approach allows us to build formal models which are consistent with our experiments. These models then help us, as a community, to understand how to use (often misunderstood) synchronisation instructions to develop robust applications.

## BODY

*Inspired by Descartes' methodic doubt, we systematically test manycore chips to dispel and correct common false memory ordering assumptions.*

## REFERENCES

- [1] J. Alglave, M. Batty, A. F. Donaldson, G. Gopalakrishnan, J. Ketema, D. Poetzl, T. Sorensen, and J. Wickerson. GPU concurrency: weak behaviours and programming assumptions. *ASPLOS*, 2015.
- [2] J. Alglave, L. Maranget, and M. Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *TOPLAS*, 2014.
- [3] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, 2010.

*Volume 3 of Tiny Transactions on Computer Science*

This content is released under the Creative Commons Attribution-NonCommercial ShareAlike License. Permission to make digital or hard copies of all or part of this work is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.  
CC BY-NC-SA 3.0: <http://creativecommons.org/licenses/by-nc-sa/3.0/>.