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

Jade Alglave¹,²  Luc Maranget³  Daniel Poetzl⁴  Tyler Sorensen¹
¹ University College London  ² Microsoft Research
³ INRIA  ⁴ University of Oxford

ABSTRACT
As a community, we (programmers, compiler writers, hardware architects, ...) often take folklore,
e.g. claims in programming guides, for granted. Inspired by Descartes’ methodic doubt, i.e. chal-
lenging 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, e.g. Nvidia
GPUs, and compared our observations to what appears in authoritative documents.

To illustrate our approach, we passed the current paragraph to a program which concureently
ciphers, then deciphers, a piece of text on a graphics processing unit (GPU). It uses a mutex, i.e.
mutable exclusion mechanism, given in the popular equational 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; however, 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
and J. Wickerson. GPU concurrency: weak behaviours and programming assumptions.
ASPLOS, 2015.
data mining for weak memory. TOPLAS, 2014.