Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Original software publication

ESBMC-GPU
A context-bounded model checking tool to verify CUDA programs

Felipe R. Monteiro [a,*], Erickson H. da S. Alves [a,b], Isabela S. Silva [a], Hussama I. Ismail [a], Lucas C. Cordeiro [a,c], Eddie B. de Lima Filho [a,d]

[a] Faculty of Technology, Federal University of Amazonas, Brazil
[b] Samsung Research Institute, Brazil
[c] Department of Computer Science, University of Oxford, United Kingdom
[d] TPV Technology Limited, Brazil

ARTICLE INFO

Article history:
Received 19 February 2017
Received in revised form 17 August 2017
Accepted 13 September 2017
Available online xxxx

Keywords:
GPU verification
Formal verification
Model checking
CUDA

ABSTRACT

The Compute Unified Device Architecture (CUDA) is a programming model used for exploring the advantages of graphics processing unit (GPU) devices, through parallelization and specialized functions and features. Nonetheless, as in other development platforms, errors may occur, due to traditional software creation processes, which may even compromise the execution of an entire system. In order to address such a problem, ESBMC-GPU was developed, as an extension to the Efficient SMT-Based Context-Bounded Model Checker (ESBMC). In summary, ESBMC processes input code through ESBMC-GPU and an abstract representation of the standard CUDA libraries, with the goal of checking a set of desired properties. Experimental results showed that ESBMC-GPU was able to correctly verify 85% of the chosen benchmarks and it also overcame other existing GPU verifiers regarding the verification of data-race conditions, array out-of-bounds violations, assertive statements, pointer safety, and the use of specific CUDA features.

* Corresponding author.
E-mail address: rms.felipe@gmail.com (F.R. Monteiro).

http://dx.doi.org/10.1016/j.scico.2017.09.005
0167-6423/

Please cite this article in press as: F.R. Monteiro et al., ESBMC-GPU A context-bounded model checking tool to verify CUDA programs, Sci. Comput. Program. (2017), http://dx.doi.org/10.1016/j.scico.2017.09.005

Software metadata

(Executable) Software metadata description	
Current software version	2.0
Permanent link to executables of this version	https://esbmc.org/gpu/ScienceofComputerProgramming/SCICO-D-17-00062
Legal Software License	Apache v2.0
Computing Operating System	Ubuntu Linux OS
Installation requirements & dependencies	GNU Libtool; Automake; Flex & Bison; Boost C++ Libraries; Multi-precision arithmetic library developers tools (libgmp3-dev package); SSL development libraries (libssl-dev package); CLang 3.8; LLDB 3.8; GNU C++ compiler (multilib files); libc6 and libc6-dev packages
Link to user manual	http://esbmc.org/gpu/ScienceofComputerProgramming/SCICO-D-17-00062
Support email for questions	lucas.cordeiro@cs.ox.ac.uk

Code metadata

Code metadata description	
Current code version	*v*2.0
Permanent link to code/repository used for this code version	https://github.com/ssvlab/esbmc-gpu/ScienceofComputerProgramming/SCICO-D-17-00062
Legal Code License	GNU Public License
Code versioning system used	git
Software code languages, tools, and services used	C++
Compilation requirements, operating environments & dependencies	GNU Libtool; Automake; Flex & Bison; Boost C++ Libraries; Multi-precision arithmetic library developers tools (libgmp3-dev package); SSL development libraries (libssl-dev package); CLang 3.8; LLDB 3.8; GNU C++ compiler (multi-lib files); libc6 and libc6-dev packages
Link to developer documentation	http://esbmc.org/gpu/ScienceofComputerProgramming/SCICO-D-17-00062
Support email for questions	lucas.cordeiro@cs.ox.ac.uk

1. Introduction

The Compute Unified Device Architecture (CUDA) is a development framework that makes use of the architecture and processing power of graphics processing units (GPUs) [1]. Indeed, CUDA is also an application programming interface (API), through which a GPU's parallelization scheme and tools can be accessed, with the goal of executing kernels [1]. Nonetheless, source code is still written by human programmers, which may result in arithmetic overflow, division by zero, and other violation types. In addition, given that CUDA allows parallelization, problems related to the latter can also occur, due to thread scheduling [2].

In order to address the mentioned issues, an extension to the Efficient SMT-Based Context-Bounded Model Checker (ESBMC) [3] was developed, named as ESBMC-GPU [4–6], with the goal of verifying CUDA-based programs (available online at http://esbmc.org/gpu/). ESBMC-GPU consists of an extension for parsing CUDA source code (*i.e.*, a front-end to ESBMC) and a CUDA operational model (COM), which is an abstract representation of the standard CUDA libraries (*i.e.*, the native API) that conservatively approximates their semantics.

A distinct feature of ESBMC-GPU, when compared with other approaches [2,7–9], is the use of Bounded Model Checking (BMC) [10] allied to Satisfiability Modulo Theories (SMT) [11], with explicit state-space exploration [12,3]. In summary, concurrency problems are tackled, up to a given loop/recursion unwinding and context bound, while each interleaving itself is symbolically handled; however, even with BMC, state-space exploration may become a very time-consuming task, which is alleviated through state hashing and Monotonic Partial Order Reduction (MPOR) [13]. As a consequence, redundant interleavings are eliminated, without ignoring a program's behavior.

Finally, existing GPU verifiers often ignore some aspects related to memory leak, data transfer, and overflow, which are normally present in CUDA programs. The proposed approach, in turn, explicitly addresses them, through an accurate checking procedure, which even considers data exchange between main program and kernel. Obviously, it results in higher verification times, but more errors can then be identified and later corrected, in another development cycle.

Existing GPU Verifiers. In addition to ESBMC-GPU, there are other tools able to verify CUDA programs and each one of them uses its own approach and targets specific property violations. For instance, GPUVerify [2] is based on synchronous, delayed visibility semantics, which focuses on detecting data race and barrier divergence, while reducing kernel verification procedures for the analysis of sequential programs. GPU+KLEE (GKLEE) [8], in turn, is a concrete plus symbolic execution tool, which considers both *kernels* and *main* functions, while checking deadlocks, memory coalescing, data race, warp divergence, and compilation level issues. In addition, Concurrency Intermediate Verification Language (CIVL) [9], a framework for static analysis and concurrent program verification, uses abstract syntax tree and partial order reduction to detect user-specified assertions, deadlocks, memory leaks, invalid pointer dereference, array out-of-bounds, and division by zero.

In fact, ESBMC-GPU differs from the aforementioned approaches due to its combination of techniques to prune the state-space exploration (*i.e.*, two-thread analysis, state hashing, and MPOR) with COM, which demonstrated effectiveness in

Please cite this article in press as: F.R. Monteiro et al., ESBMC-GPU A context-bounded model checking tool to verify CUDA programs, Sci. Comput. Program. (2017), http://dx.doi.org/10.1016/j.scico.2017.09.005