Evaluation of Graphics-based General Purpose Computation Solutions for Safety Critical Systems: An Avionics Case Study

Marc Benito*
Matina Maria Trompouki
Leonidas Kosmidis
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catalunya (UPC)

Juan David Garcia
Sergio Carretero
Airbus Defence and Space
Getafe, Spain

Ken Wenger
CoreAVI
Waterloo, ON, Canada

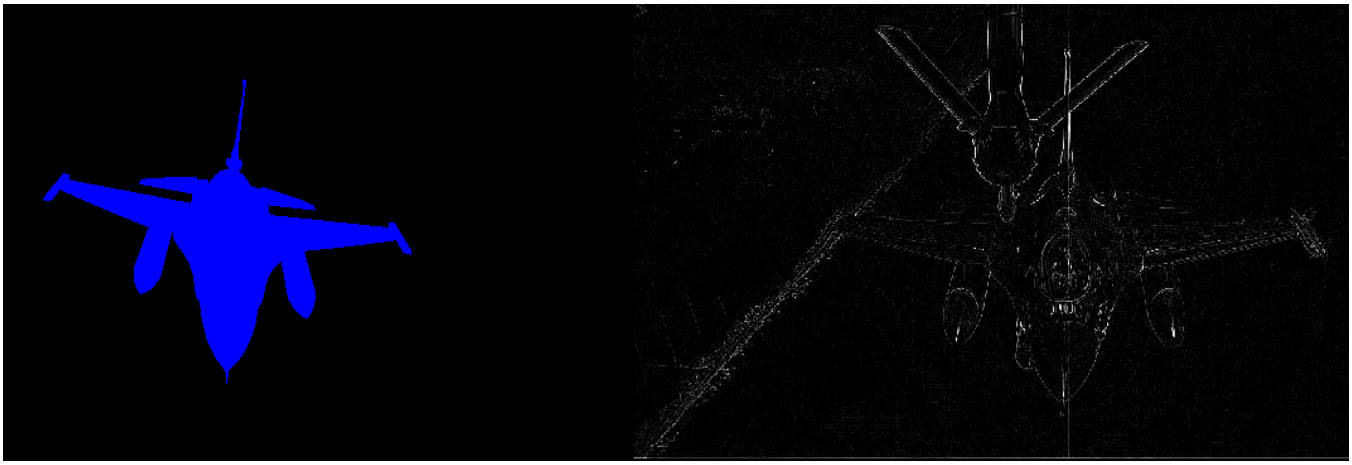

Figure 1: Prototype Airbus application which was ported and evaluated in OpenGL SC 2 and Brook SC.

1 INTRODUCTION

Current safety-critical systems require increased performance to support advanced functionalities such as autonomous operation. Graphics Processing Units (GPUs) can provide this required performance, however their general purpose programming models like CUDA or OpenCL cannot be used, since they are based on programming features not allowed in safety-critical systems like pointers and dynamic memory allocation [Trompouki and Kosmidis 2018]. This prevents their software certification according to safety standards such as ISO26262 (automotive) and DO-178B (avionics).

2 OUR APPROACH

In order to allow the certification of general purpose GPU software in critical systems, two solutions currently exist. First, OpenGL SC 2 is an OpenGL subset designed for safety critical systems. Although directly programming general purpose algorithms on graphics is possible, it is very complex and error prone. The second solution, the open source technology Brook SC [Leonidas Kosmidis, Matina Maria Trompouki et al 2018], based on [Trompouki and Kosmidis 2018], allows to program safety-critical applications in a CUDA-like high-level general purpose GPU language, Brook, which does not allow pointers and dynamic memory allocation and is transformed into OpenGL SC 2, retaining the advantage of the GPU code certification. Moreover, unlike closed-source toolchains like CUDA and OpenCL with massive code-bases, the small size of Brook SC and its open source nature simplifies the collection of the required evidence to support its tool qualification, as studied in [Trompouki and Kosmidis 2019] for ISO 26262.

In this work, we compare both solutions in terms of performance and programming productivity, using an avionics application running on a safety-critical GPU software and hardware, consisting of a commercial, certified OpenGL SC 2 driver and an AMD E8860 avionics-grade GPU. In this Bachelor's thesis project [Benito 2019], awarded with a 2019 HiPEAC Technology Transfer Award, an Airbus prototype application shown in Figure 1, written in Vulkan and performing general-purpose computations was ported to both solutions. The application consists of a graphics part (left) and general purpose computation (right), processing images from a camera.

The porting to Brook SC was completed in record time (3 days) versus 3 weeks required for the OpenGL SC 2 port, without any prior knowledge of either solution. This difference is reflected in the fact that the Brook SC port requires an order of magnitude less lines of code (LOC), 160 compared to the 1200 LOC required for the OpenGL SC 2 version. In terms of performance, both implementations are equivalent, achieving image processing rate of 60 frames per second (fps) as the original application. However, when vsync is disabled the OpenGL SC 2 achieves 2800 fps and the Brook SC 2500 fps, which represents a minor overhead for a massive leap in programmability and productivity compared to direct OpenGL SC 2 development.

REFERENCES

Marc Benito. 2019. *Analysis and Evaluation of Embedded Graphics Solutions for Critical Systems.* Bachelor's thesis. Universitat Politècnica de Catalunya (UPC).

Leonidas Kosmidis, Matina Maria Trompouki et al. 2018. Brook SC. http://github.com/lkosmid/brook.

Matina Maria Trompouki and Leonidas Kosmidis. 2018. Brook Auto: High-Level Certification-Friendly Programming for GPU-Powered Automotive Systems. In *Proceedings of the 55th Annual Design Automation Conference (DAC '18).*

Matina Maria Trompouki and Leonidas Kosmidis. 2019. BRASIL: A High-Integrity GPGPU Toolchain for Automotive Systems. In *Proceedings of the 38th IEEE International Conference on Computer Design (ICCD '19).*

*{name.surname}@bsc.es